دروس لغة الأسمبلي التابعة لموقع الفريق العربي للبرمجة
http://www.arabteam2000.com

ملاحظة:
هذه الدروس هي بالأساس مقتبسة من منهاج السنة الثانية قسم هندسة الحاسبات بجامعة حلب

الجزء الأول
مقدمة في لغة الأسمبلي

لمحة عن أنظمة العد

تمهيد

اعتاد الإنسان على نظام العد العشري لأنه كان يملك عشرة أصابع في يديه، فعندما يريد إحصاء الأشياء أمامه فكان يقابل كل عنصر من الموجودات أمامه بإصبع واحدة من يديه، و عندما تنتهي أصابع يديه فإنه يحتاج إلى شخص آخر يرفع إصبع واحدة حيث تمثل كل إصبع من أصابع الشخص الثاني عشرة أصابع من أصابع الشخص الأول و بذلك كان الثاني يلعب دور العشرات أما الأول فيلعب دور الآحاد.

و بعد اختراع الكتابة سارع علماء الرياضيات إلى تحويل نظام العد العشري إلى صيغة كتابية، فاعتمدوا الأساس التالي: (تمثل الأعداد من 1 حتى 9 برمز واحد فقط أما العدد الذي يأتي بعد التسعة فهو عبارة عن مزيج رمزين الأول هو الصفر و الثاني هو الواحد).

من الفكرة السابقة نجد أن الرموز الأساسية لنظام العد العشري هي من الصفر حتى التسعة أي هي عشرة رموز نستطيع من خلالها تكوين عدد أي عدد طبيعي.

طريقة العد:

نبدأ بالعد اعتباراً من أول رمز و هو الصفر و نزيد بمقدار واحد واحد إلى أن نصل إلى نهاية الرموز ألا و هو التسعة، و إذا أردنا المتابعة فإننا نصفر الخانة التي نتعامل معها و نضيف واحد إلى الخانة المجاورة لنحصل على الرقم عشرة (10) و من ثم نبدأ بزيادة الآحاد من جديد حتى نصل إلى 19 عندها نصفر الآحاد و نضيف واحد إلى خانة العشرات فينتج العدد 20 و هكذا حتى نصل إلى العدد 99 عندها نحاول زيادة خانة الآحاد فلا نستطيع فنصفرها و نحاول زيادة العشرات فلا نستطيع أيضاً فنصفرها و نزيد خانة إلى منزلة المئات فنحصل على العدد 100.

العد بالنظام الست عشري

يختلف هذا النظام عن سلفه بأن الرموز الأساسية هي من الصفر حتى التسعة و يأتي بعد التسعة الأحرف من A حتى F أي أن الرموز الأساسية هي:

{ 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F }

و لكي تستطيع العد بسهولة في هذا النظام أعد قراءة التمهيد و لكن تخيل جدلاً أن للإنسان ست عشرة إصبع في كل يد ثمانية أصابع !!

تمرين على العد بالنظام الست عشري:

0,1,2,…,9,A,B,…,F,10,11,12,13,14,…,19,1A,1B,1C,…,1F,20,21,…,29,2A,2B,…,2F,30,…,99,9A,9B,…,9F,A0,A1,A2,…,A9,AA,AB,AC,…,AF,…,FF,100,…,119,11A,11B,…,199,19A,…

نظام العد الثنائي

تتطلب أجهزة الحواسيب و الأجهزة الإلكترونية نظام عد جديد ملائم لطبيعة هذه الأجهزة، فنحن نعلم أن جميع الأجهزة الإلكترونية تعمل على التيار الكهربائي و الذي له حالتين هما الوضع on و الوضع off .

و بذلك كان النظام الثنائي هو الحل حيث اعتمد على رمزين فقط في تمثيل أعداده هما الصفر و الواحد {0,1} .

العد بالنظام الثنائي:

0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110,1111

التحويل بين نظم الأعداد

يلزمنا في لغة الأسمبلي التحويلات التالية:

1- التحويل من الثنائي إلى العشري.

2- التحويل من الست عشري إلى العشري.

3- التحويل من العشري إلى الثنائي.

و سنعطي مثالاً عن كل حالة من هذه الحالات:

مثال1 : حول الرقم الثنائي التالي 0100 إلى مقابله في نظام العد العشري:

$$(0100)b = 0\times 2^0 + 0\times 2^1 + 1\times 2^2 + 0\times 2^3 = 0+0+4+0 = 4$$

مثال2 :حول العدد الست عشري التالي 33A إلى مقابله في نظام العد العشري:

$$(33A)h = 10\times 16^0 + 3\times 16^1 + 3\times 16^2 = 10+48+768 = 826$$

مثال3 : حول العدد العشري التالي 30 إلى مقابله في النظام الثنائي:

لدينا الجدول المرسوم جانباً:

...	128	64	32	16	8	4	2	1

نستخدم هذا الجدول من أجل هذا النوع من التحويل فلتحويل العدد العشري 30 نلاحظ أنه مكون من 2+4+8+16 فنضع واحدات تحت الأعداد 16 و 8 و 4 و 2 و نملأ الباقي أصفاراً، و بذلك نحصل على الرقم الثنائي المقابل.

المتمم الثنائي و كيفية الحصول عليه

يستخدم المتمم الثنائي من أجل تمثيل الأعداد السالبة في الحاسب في النظام الثنائي و لتمثيل عدد سالب نتبع الخطوات التالية:

1- نكتب العدد بالنظام الثنائي.

2- نقلب الأصفار واحدات و الواحدات أصفاراً.

3- نضيف واحد إلى الرقم الناتج.

مثال: مثّل العدد 30- بالنظام الثنائي عن طريق المتمم الثنائي:

(30)d = 0001 1110
نقلب => 1110 0001
نضيف => 1110 0010

لمحة عن الحاسب

يُعرَّف الحاسب الرقمي بأنه نظام إلكتروني لمعالجة المعطيات، و يتألف من قسمين أساسيين:

القسم الأول : البرمجيات SOFTWARE .

القسم الثاني : الكيان الصلب HARDWARE

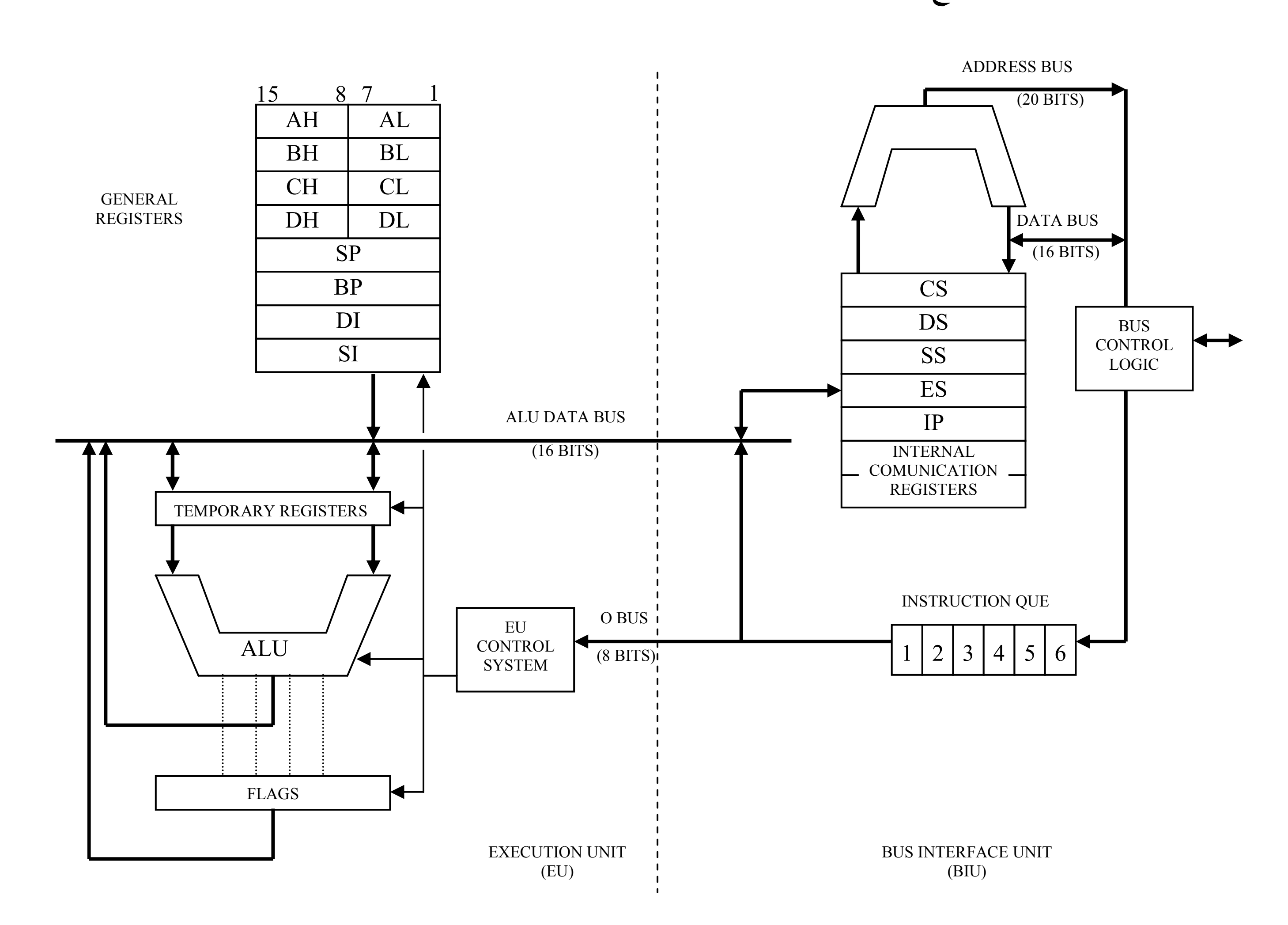
و يقسم الكيان الصلب إلى أقسام رئيسية هي :

وحدة الذاكرة

وحدة الدخل → C.P.U → وحدة الخرج

1) وحدة الدخل: تتم من خلالها إدخال المعطيات الرقمية.
2) وحدة الخرج: تتم من خلالها إظهار النتائج بعد معالجة المعطيات.
3) وحدة المعالجة المركزية: هي المسؤولة عن العمليات الحسابية و المنطقية و معالجة البيانات.
4) وحدة الذاكرة: تخزن البرامج و المعطيات.

البنية الداخلية للمعالج 8086



يتألف المعالج 8086 من وحدتين منفصلتين هما :

1) وحدة ملائمة الممرات (Execution Unit) : و سنرمز لها بالرمز EU .
2) وحدة التنفيذ (Bus interface Unit) : و سنرمز لها بالرمز BIU .

بشكل عام فإن الـــ BIU مسؤولة عن معظم الأعمال مثل : إحضار التعليمة، قراءة و كتابة المتحولات في الذاكرة، إدخـــال و إخراج المعطيات من و إلى الأجهزة المحيطية.

أما الـــ EU فهي مسؤولة عن تنفيذ التعليمات. و كلا الوحدتين تعملان بشكل متوازٍ لتخفيض الزمن المطلوب لإحضار عـــدة تعليمات و تنفيذها.

ملاحظة: من الجدير بالذكر بأن هنالك ثلاثة ممرات في الحاسب و هي:

1) ممر المعطيات DATA BUS : و يصل بين المعالج و الذاكرة وظيفته نقل المعطيات من و إلى الذاكرة.
2) ممر العناوين ADDRESS BUS : و يصل بين المعالج و الذاكرة أيضاً و وظيفته نقل العناوين من المعالج إلى الذاكرة.
3) ممر التحكم CONTROL BUS : لتنسيق عمل الممرين السابقين.

وحدة ملائمة الممرات Bus Interface Unit

و تستخدم لملائمة المعالج مع العالم الخارجي. و تتألف من : جامع العناوين، مسجلات المقاطع، وحدة التحكم بالمحرف، صف التعليمات.

تقوم وحدة الـــ BIU بالتحكم بممر المعطيات و ممر العناوين و ممر التحكم .

تحضر BIU التعليمات من الذاكرة بايت بايت و تضعها فيما يسمى برتل التعليمات (صف التعليمات) الذي يتسع لست بايتات كحد أعظمي و من الطبيعي أن التعليمة التي تدخل رتل التعليمات أولاً يتم تنفيذها أولاً للمحافظة على ترتيب التعليمـــات و يدعى هذا المبدأ بـــ الداخل أولاً خارج أولاً First In Last Out و نرمز لهذا المبدأ بـــ FIFO.

إن إحضار شيفرة التعليمة التالية يتم عندما تكون وحدة التنفيذ EU مشغولة بتنفيذ التعليمة الحالية (هذه إحدى محسنات المعالج 8086 عن أسلافه حيث كانت الـــ CPU في المعالجات السابقة للمعالج 8086 تتوقف عن العمل خلال فترة تنفيذ التعليمة الحالية).

عندما تفك وحدة التنفيذ EU شيفرة تعليمة ما من رتل التعليمات و تكون هذه التعليمة تعليمة تؤدي إلى تغيير تسلسل تعليمات البرنامج (قفز إلى برنامج فرعي مثلاً) عندها يتم تصفير رتل التعليمات و إعادة ملئه من جديد بتعليمات البرنامج الفرعي (لأن وحدة ملائمة الممرات BIU تجلب التعليمات دون معرفة ما تؤديه هذه التعليمات).

ملاحظة: جامع العناوين و مسجلات المقاطع سيتم شرحها لاحقاً.

وحدة التنفيذ Execution Unit

و هي مسؤولة عن فك شيفرة التعليمات و تنفيذها و تتألف من :

1) و حدة الحساب و المنطق.
2) مسجل الأعلام.
3) ثمانية مسجلات للأغراض العامة.
4) مسجلات مؤقتة.
5) منطق التحكم بـــ EU.

تجلب وحدة التنفيذ EU التعليمات من مقدمة رتل التعليمات في وحدة ملائمة الممرات BIU و تفك شيفرتها و تقوم بالعمل الذي تمليه كل تعليمة فإذا احتاجت هذه الوحدة (EU) إلى معلومة مخزنة في الذاكرة فإنها تأمر وحدة ملائمة الممرات BIU بإحضارها و ذلك عن طريق إعطائها عنوان هذه المعلومة في الذاكرة.

إن من أحد أهم وظائف EU هو تنفيذ العمليات الحسابية و المنطقية على المعلومات، و أثناء سير التنفيذ تقوم EU بفحص مسجل الأعلام بعد كل تعليمة (مسجل الأعلام : هو عبارة عن ستة عشر بت تعبر عن حالة المعالج بعد تنفيذ كل تعليمة) .

مسجلات الأغراض العامة هي ثمانية مسجلات طول كل مسجل منها 2 بايت و هذه المسجلات هي AX,BX,CX,DX,SI,DI,BP,SP .

بنية الذاكرة

تتألف الذاكرة من حجرات متسلسلة سعة كل منها 8 بت (واحد بايت) ، ترقم هذه الحجرات من الصفر و حتى نهاية الذاكرة و يستخدم النظام الست عشري عادة في عملية الترقيم و بذلك يكون لكل حجرة رقم يميزها عن غيرها، يدعى هذا الرقم بعنوان تلك الحجرة.

يوضع داخل كل حجرة رقم ست عشري يتراوح بين 0 و FF و يدعى هذا الرقم بمحتوى تلك الحجرة.

يوجد بين المعالج و الذاكرة ممران هما ممر المعطيات بعرض 16 بت و ممر العناوين بعرض 20 بت.

فمثلاً عندما يحتاج المعالج إلى القيمة المخزنة في الحجرة ذات الرقم 100 (عنوانها 100) فإن الرقم 100 يمثل بشكل ثنائي و يوضع على ممر العناوين و يرسل إلى الذاكرة، و حالما تستلم الذاكرة هذا العنوان فإن محتوى الحجرة 100 يرسل إلى المعالج عن طريق ممر المعطيات.

إن كون ممر العناوين ذو عرض 20 بت (20 خط نقل) هذا يعني أنه يستطيع نقل رقم ثنائي ذو 20 خانة أي أن أكبر قيمة يمكن وضعها على ممر العناوين هي :

$$2^{20} = 1048576 \approx 1MB$$

و بذلك يستطيع المعالج 8086 عنونة واحد ميغا من الذاكرة فقط.

مقاطع الذاكرة (هذه الفقرة مرتبطة ارتباطاً وثيقاً بالمسجلات)

يتعامل المعالج كما ذكرنا مع واحد ميغا من الذاكرة، و يمكن أن نقتطع من هذه الميغا أربعة مقاطع أساسية يتعامل معها برنامجنا بشكل مباشر (أي أنه لا تتم الاستفادة من كل الذاكرة بآن واحد) و هذه المقاطع الأربعة هي:

1) مقطع الشيفرة Code Segment CS

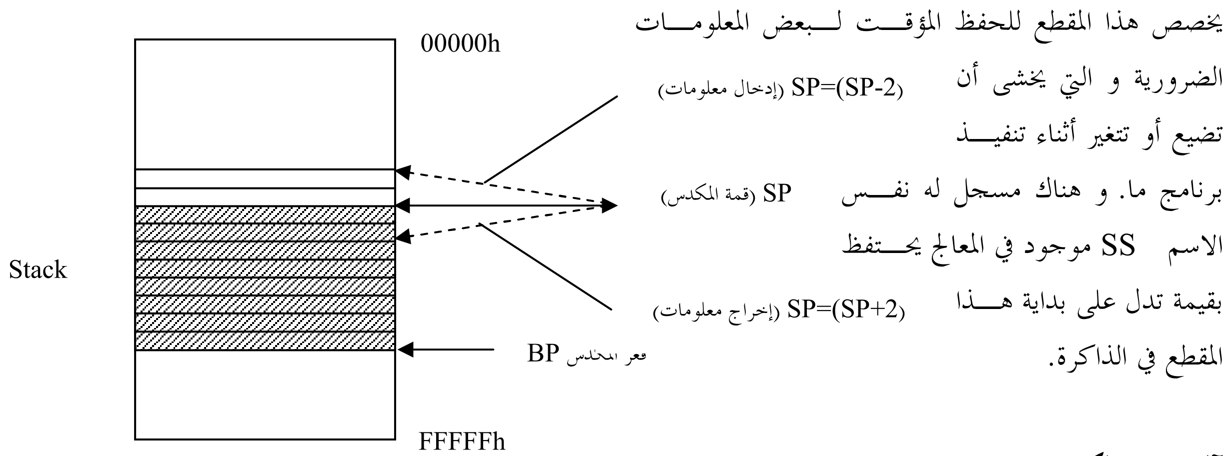
يخصص هذا المقطع من الذاكرة –كما هو واضح من تسميته– لتخزين شيفرة البرنامج. و هناك مسجل له نفس الاسم CS موجود في المعالج يحتفظ بقيمة تدل على بداية هذا المقطع في الذاكرة و يساعده المسجل (Instruction Pointer) IP الذي يحتفظ بعنوان التعليمة التي ستنفذ الآن و تعدل قيمته آلياً ليشير إلى عنوان التعليمة التالية.

2) مقطع المعطيات Data Segment DS

يخصص هذا المقطع من الذاكرة لتخزين المعطيات و المتحولات. و هناك مسجل له نفس الاسم DS موجود في المعالج يحتفظ بقيمة تدل على بداية هذا المقطع في الذاكرة و يساعده المسجل SI الذي يشير إلى الإزاحة بالنسبة إلى بدايته.

3) مقطع المكدس Stack Segment SS

يخصص هذا المقطع للحفظ المؤقت لبعض المعلومات الضرورية و التي يخشى أن تضيع أو تتغير أثناء تنفيذ برنامج ما. و هناك مسجل له نفس الاسم SS موجود في المعالج يحتفظ بقيمة تدل على بداية هذا المقطع في الذاكرة.



آلية عمل المكدس Last In First Out LIFO (آخر ما يدخل أول ما يخرج) : أي أن أول عنصر يدخل إلى المكدس يصبح في قعره و آخر عنصر يدخل المكدس يصبح في قمته و يتم سحب المعلومات من المكدس من قمته حيث لدينا مسجل اسمه Stack Pointer SP يشير دوماً إلى قمة المكدس فهو يتغير حسب الحالة التي يتم بها التعامل مع المكدس (إدخال معلومات أو إخراج). فعند إدخال معلومة بطول 2 بايت فإن قمة المكدس تقترب من بداية الذاكرة (انظر الشكل) و بذلك تنقص قيمة SP بمقدار 2 **لأن إملاء المكدس يعني الاقتراب من العنوان الأصغر** و العكس بالعكس أي عندما نسحب معلومة من المكدس فإن قمته تبتعد عن بداية الذاكرة و بذلك تزيد SP بمقدار 2 **لأن إفراغ المكدس يعني الاقتراب من العنوان الأكبر**.

4) مقطع المعطيات الإضافي Extra Segment ES

يستخدم عند الحاجة إلى استخدام مقطعي معطيات بنفس الوقت و بذلك نستطيع الاستفادة من مساحة أكبر في الذاكرة. و يساعده المسجل Destination Index DI الموجود في المعالج و الذي يشير إلى الإزاحة بالنسبة إلى بدايته.

ملاحظة: يجب التمييز بين المقطع و مسجل المقطع حيث المقطع هو جزء من الذاكرة بينما مسجل المقطع يتألف من بايتين و هو موجود في المعالج.

المسجلات Registers

يملك المعالج 8086 أربعة مجموعات من المسجلات ذات 16 بت يستطيع المبرمج الوصول إليها و هي:

1) مؤشر التعليمة IP
2) أربعة مسجلات معطيات AX,BX,CX,DX .
3) أربعة مسجلات تأشير و فهرسة SI,DI,BP,SP .
4) أربعة مسجلات مقاطع CS,DS,SS,ES .

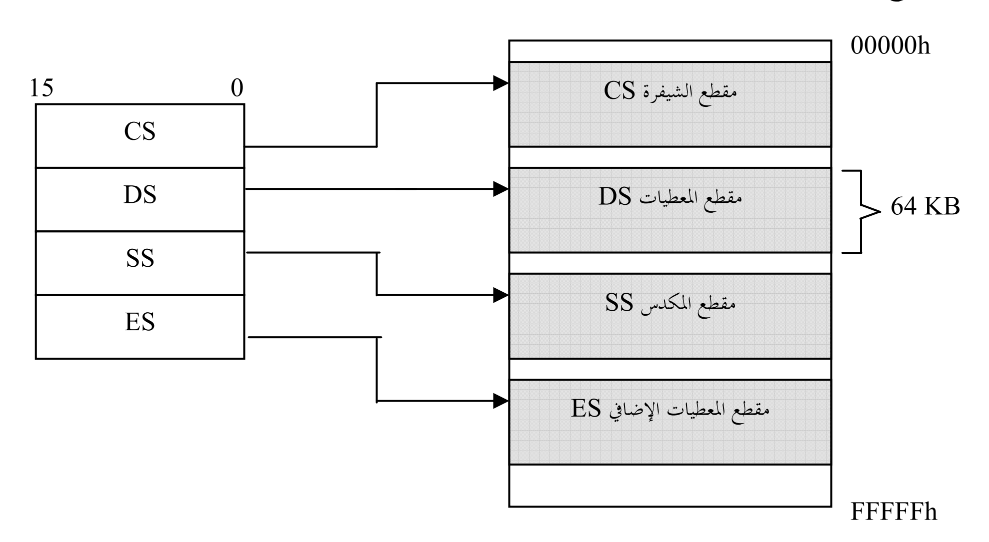
بالإضافة إلى ذلك يوجد مسجل آخر هو مسجل الأعلام و يدعى أيضاً مسجل الحالة و هو مسجل ذو 16 بت و لكن نستخدم منه 9 خانات فقط.

سنشرح كلٍ من هذه المسجلات بالتفصيل :

المجموعة الأولى : مسجلات المقاطع

و هي عبارة عن أربعة مسجلات طول كل منها 16 بت أي 2 بايت و هي :

1) مسجل مقطع الشيفرة CS : يحتوي على عنوان أول حجرة في مقطع شيفرة البرنامج في الذاكرة، أي أنه يشير إلى بداية مقطع الشيفرة.
2) مسجل مقطع المعطيات DS : يحتوي على عنوان أول حجرة في مقطع المعطيات في الذاكرة، أي أنه يشير إلى بداية مقطع المعطيات.
3) مسجل مقطع المكدس SS : يحتوي على عنوان أول حجرة في مقطع المكدس في الذاكرة، أي أنه يشير إلى بداية مقطع المكدس.
4) مسجل مقطع المعطيات الإضافي ES : يحتوي على عنوان أول حجرة في مقطع المعطيات الإضافي في الذاكرة، أي أنه يشير إلى بداية مقطع المعطيات الإضافي.



المجموعة الثانية: مسجلات الفهرسة و التأشير

و هي عبارة عن أربعة مسجلات مساعدة تساعد في إيجاد العنوان الفيزيائي بالتعاون مع مسجلات المقاطع، و طول هذه المسجلات 16 بت أي 2 بايت، و هي :

1) مسجل دليل المصدر Source Index SI : يخزن فيه عنوان يدل على الإزاحة ضمن مقطع المعطيات DS و بمعنى آخر يستعمل في إمساك العناوين الفعالة من أجل التعليمات التي تتناول المعطيات المخزنة في مقطع المعطيات في الذاكرة.
2) مسجل دليل الهدف Destination Index DI : يخزن فيه عنوان يدل على الإزاحة ضمن مقطع المعطيات الإضافي ES ، و بمعنى آخر يستعمل مسجل دليل الهدف DI من أجل استنتاج العنوان الفيزيائي الذي يحدد حجرة متحول الهدف.
3) مسجل مؤشر المكدس Stack Pointer SP : يسمح مؤشر المكدس بوصول سهل للحجرات في مقطع المكدس الموجود في الذاكرة حيث أن القيمة في SP تمثل العنوان الفعال لحجرة المكدس التالية التي يمكن الوصول إليها نسبة إلى العنوان الحالي الموجود في مسجل مقطع المكدس SS و يحتفظ SP دوماً بقيمة تدل على قمة المكدس ، هذا و إن قيمة هذا المسجل تتعدل تلقائياً عند وضع أو سحب معلومة بالمكدس.

4) مسجل مؤشر القاعدة Base Pointer BP : يحوي قيمة تدل على الإزاحة بالنسبة لمقطع المكدس SS و هو يستخدم لقراءة المعطيات ضمن مقطع المكدس بدون إزالتها من المكدس.

المجموعة الثالثة: مسجلات المعطيات

تستخدم هذه المسجلات من أجل التخزين المؤقت للنتائج المرحلية أثناء تنفيذ البرنامج حيث أن تخزين المعطيات في هذه المسجلات يمكننا من الولوج إلى تلك المعطيات بشكل أسرع مما لو كانت في الذاكرة، و تقسم المسجلات إلى :

1) مسجل المراكم Accumulator و يرمز له بالرمز A .
2) مسجل القاعدة Base و يرمز له بالرمز B .
3) مسجل العد Count و يرمز له بالرمز C .
4) مسجل المعطيات Data و يرمز له بالرمز D .

و كل مسجل من المسجلات السابقة يمكن استعماله إما ككلمة 16 بت و يدل على ذلك بكتابة الحرف X بعد اسم المسجل أو يمكن استعماله كبايتين كل منهما 8 بت و يدل على ذلك باستخدام الحرفين H,L حيث :

L للبايت ذو العنوان الأصغر ، مثال AL .

H للبايت ذو العنوان الأكبر ، مثال BH .

هكذا و إن كلاً من هذه المسجلات يمكن استخدامه من أجل التعليمات الرياضية أو المنطقية في لغة الأسمبلي مثل And, Add .

و من أجل بعض التعليمات مثل البرامج التي تحتوي على تعليمات سلاسل فإنها تستعمل مسجلات معينة مثل استعمال المسجل C لتخزين العدد الذي يمثل عدد البايتات التي ستنفذ عليها تعليمات السلاسل (عدد مرات تكرار تعليمة السلسلة)

مسجل مؤشر التعليمة Instruction Pointer IP

هذا المسجل يحدد موقع التعليمة التالية التي ستنفذ في مقطع الشيفرة و بعد جلب شيفرة التعليمة من الذاكرة فإن BIU تعدل قيمة IP بحيث تشير إلى التعليمة التالية في الذاكرة (التعديل يتم آلياً).

مسجل الأعلام Flags Register

هو مسجل ذو 16 بت موجود في وحدة التنفيذ كما هو واضح بالشكل :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

و كما نلاحظ من الشكل السابق أنه يوجد ستة أعلام للحالة هي CF, PF, AF, ZF, SF, OF ، و كذلك يوجد ثلاثة أعلام للتحكم DF, IF, TF .

أ) أعلام الحالة

تشير إلى الحالات الناتجة كنتيجة لتنفيذ تعليمة منطقية أو رياضية حيث تكون إما في حالة واحد منطقي Set أو تكون في حالة صفر منطقي Reset ، و سنلخص فيما يلي عمل كلٍ منها:

أولاً: علم الإنزياح Carry Flag

يكون في حالة الواحد المنطقي إذا وجد انزياح خارجي (حمل) أو استعارة من أجل الخانة الأخيرة (البت الأخير) و ذلك أثناء تنفيذ التعليمات الرياضية.

و يكون في حالة الصفر المنطقي إذا لم يوجد حمل أو استعارة من أجل البت الأخير.

أمثلة:

أولاً: حالة الإنزياح

	7	6	5	4	3	2	1	0
	1	1	0	0	0	1	1	0
+	1	1	0	0	0	1	1	1
CF=1	1	0	0	0	1	1	0	1

لاحظ بأن النتيجة لم تتسع في ثمانية بتات و إنما تحتاج إلى تسع بتات و نعبر عن ذلك بثمانية بتات و CF=1 أي أنه لدينا في اليد واحد.

ببساطة: فمهما كَبُرَ العددان فإن تسعة بتات يمكن أن تستوعبها.

ثانياً: حالة الاستعارة

	7	6	5	4	3	2	1	0
1	0	1	0	1	1	0	1	1
–	1	1	1	1	1	0	0	0
	0	1	1	0	0	0	1	1

لاحظ بأن العدد الأول الممثل ثنائياً أصغر من العدد الثاني الممثل ثنائياً أيضاً ، لذلك فعند إجراء عملية الطرح و في مثالنا هذا تخيلنا بت تاسع فيه القيمة واحد (استعرنا) و بالتالي فإن CF=1 أي لدينا استعارة من أجل البت الأعلى رتبة.

و في المثالين السابقين نطبق نفس الكلام من أجل 2 بايت و لكن الإنزياح الخارج و الاستعارة تكون من أجل البت الخـــامس عشر (الأخير).

ثانياً: علم الازدواجية Parity Flag PF

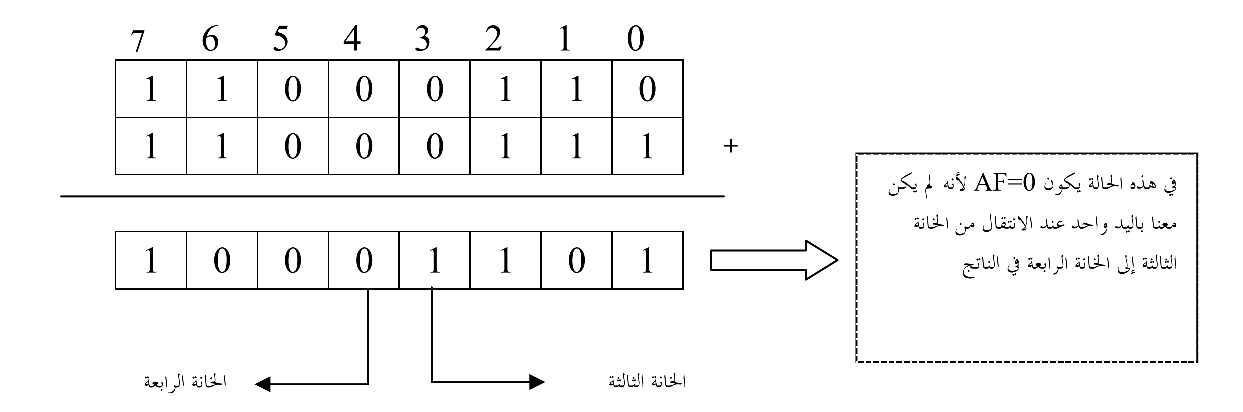
يصبح في حالة واحد منطقي إذا كانت نتيجة آخر تعليمة تحوي على عدداً زوجياً من الخانات الواحدية (بعد التحويل إلى النظام الثنائي طبعاً) و إلا يكون في حالة الصفر المنطقي.

نلاحظ أن علم PF يفحص البايت السفلي فقط حتى لو كنا نتعامل مع كلمة (2 بايت) ، أما عندما نتعامل مع بايت واحد فقط فإنه يفحصه كله.

ثالثاً: علم الإنزياح المساعد Auxiliary Flag AF

يكون في حالة الواحد المنطقي إذا وجد إنزياح من النصف السفلي إلى النصف العلوي أو استعارة من النصف العلوي إلى النصف السفلي و ذلك من أجل البايت السفلي من الكلمة (2 بايت) و بمعنى آخر أنه إذا كان لدينا إنزياح من الخانة 3 إلى الخانة 4 فإن AF=1 و ذلك في حال كانت المعطيات بايت واحد أو بايتين (كلمة)، و فيما عدا ذلك يكون AF=0 .

مثال:



رابعاً: علم الصفر Zero Flag ZF

يصبح في حالة واحد منطقي عندما يكون ناتج آخر عملية حسابية أو منطقية يساوي الصفر.

يصبح في حالة صفر منطقي عندما يكون ناتج آخر عملية حسابية أو منطقية لا يساوي الصفر.

خامساً: علم الإشارة Sign Flag SF

يكون علم SF في حالة واحد منطقي Set إذا كانت نتيجة آخر عملية حسابية عدداً سالباً.

يكون علم SF في حالة صفر منطقي Reset إذا كانت نتيجة آخر عملية حسابية عدداً موجباً.

مصطلح: من إحدى طرق تمثيل الأعداد السالبة في الكمبيوتر هي اعتبار الخانة الأخيرة مخصصة للإشارة و بما أن البايت مكـون من ثمانية خانات فسيتم اقتطاع الخانة الأخيرة منه من أجل الإشارة فإن احتوت على القيمة واحد فإن الخانات السبعة الباقية هي عدد ثنائي سالب أما إذا احتوت على القيمة صفر فإن الخانات السبعة المتبقية ما هي إلا عدد موجب.

و بذلك يكون SF هو نسخة عن الخانة الأخيرة في الناتج عند اعتماد هذا النظام لتمثيل الأعداد السالبة.

لاحظ أنه انطلاقاً من هذا المبدأ في التمثيل يمكننا تمثيل المجالات التالية من الأعداد:

من أجل بايت واحد من –128 إلى +127

من أجل بايتين من –32768 إلى +32767

سادساً: علم الطفحان Overflow Flag OF

يكون في حالة واحد منطقي عندما لا تتسع النتيجة في المكان المخصص لتخزينها أي تتجاوز القدرة التخزينية، أما إذا لم تكـن النتيجة خارج المجال المحدد فإن OF يبقى في حالة الصفر المنطقي.

يحدث الطفحان في الحالات التالية:

1) جمع أعداد موجبة كبيرة.

2) جمع أعداد سالبة كبيرة.

3) طرح عدد موجب كبير من عدد سالب كبير.

4) طرح عدد سالب كبير من عدد موجب كبير.

ملاحظة: جميع الأعلام السابقة ما عدا CF تُقرأ فقط أي لا نستطيع تغيير محتواها لذلك يمكن قراءتها فقط و لا يمكـن تغـيير محتوياتها بواسطة تعليمات برمجية مباشرة.

المعالج مزود بتعليمات تستطيع اختبار حالة هذه الأعلام لتغيير تتابع تنفيذ البرنامج فمثلاً يمكن اختبار علم ZF=1 كشرط من أجل القفز إلى جزء آخر من البرنامج.

و فيما يلي سنشرح أعلام التحكم:

أولاً: علم الخطوة الوحيدة Trap Flag TF

يوضع بالحالة واحد منطقي عندما نرغب بتنفيذ البرنامج خطوة خطوة و هو مفيد عندما نريد تصحيح برنامجنا و استكشاف مواقع الأخطاء.

ثانياً: علم المقاطعة Interrupt Flag IF

يستخدم من أجل التعبير عن إمكانية أو عدم إمكانية تنفيذ المقاطعة، فيوضع بالحالة واحد منطقي عندما لا نرغب بتنفيذ أي مقاطعة (المقاطعة محجوبة) أما عند وضعه في حالة الصفر المنطقي فإن المقاطعة مسموح بها.

ملاحظة: المقاطعة هي عبارة عن خدمة تؤدي إلى عمل معين فمثلاً المقاطعة 21 و التي من أحد خدماتها العودة إلى نظام التشغيل.

ثالثاً: علم الاتجاه Direction Flag DF

يدل على اتجاه سير العمليات التسلسلية.

عندما يكون في حالة واحد منطقي فإن السلسلة تكون من العنوان الأعلى إلى العنوان الأدنى.

عندما يكون في حالة صفر منطقي فإن السلسلة تكون من العنوان الأدنى إلى العنوان الأعلى.

مفهوم العنوان الفيزيائي و الإزاحات

مقدمة

لاحظنا أن الذاكرة بطول 1 ميغا بايت أي أنها مرقمة من 00000h إلى FFFFFh لذلك فإننا نحتاج أثناء عنونة المقاطع إلى رقم ست عشري بطول 20 بت ذلك لأن تمثيل رقم ست عشري بطول خمس خانات (و هو المستخدم في ترقيم حجرات الذاكرة) يحتاج إلى عشرين بت لكن مسجلات المقاطع و التي نستخدمها في العنونة هي بطول 16 بت فقط الأمر الذي يضطرنا إلى استنتاج عنوان فيزيائي بعشرين بت !!

آلية الحصول على العنوان الفيزيائي Physical Address PA

يلزمنا لإيجاد العنوان الفيزيائي قيمتين هما :

1) قيمة مسجل المقطع
2) قيمة المسجل المساعد له

<u>فكرة Very good Tip :</u>

عندما نريد إزاحة رقم ممثل بالنظام العشري خانة واحدة نحو اليسار فإننا نضربه بعشرة !!

مثال: هل تستطيع إزاحة الرقم 192 إلى اليسار خطوة واحدة ليصبح 1920 ؟؟

نعم و ذلك بضربه بعشرة كالتالي $192 \times 10 = 1920$

و كذلك الأمر في النظام الست عشري، فعندما نريد إزاحة رقم ست عشري فإننا نضربه بعشرة النظام الست عشري و التي هي

10 h = 16 d

↓ مقابلها في النظام العشري ↓ عشرة النظام الست عشري

لذلك يتم الحصول على العنوان الفيزيائي بالطريقة التالية:

1) نأخذ قيمة مسجل المقطع الممثلة بالنظام الست عشري و نضربها بعشرة النظام الست عشري فتزاح قيمة مسجل المقطع خانة واحدة نحو اليسار.
2) نجمع قيمة المسجل المساعد لنفس المقطع و الممثلة أيضاً بالنظام الست عشري فتكون النتيجة هي حصولنا على العنوان الفيزيائي

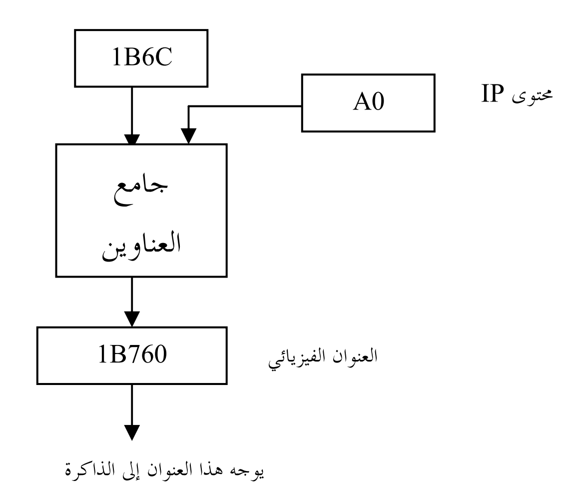
PA (Physical Address) = قيمة المسجل المساعد + (مسجل المقطع x 10h)

أمثلة:

بفرض لدينا مسجل مقطع الشيفرة CS يحتوي على 1B6C و قيمة مسجل مؤشر التعليمة IP المساعد له هي A0 أوجد العنوان الفيزيائي للتعليمة :

الحل:

PA = (CS x 10h) + IP = 1B6C x 10h + A0 = 1B760



مثال آخر: أوجد PA بفرض DS = 1000h و SI = 1F .

الحل:

PA = (1000 x 10) + 1F = 1001F

الطريقة العكسية (هذه الطريقة يجب إتقانها ذهنياً)

عندما نُعطى العنوان الفيزيائي و نريد استنتاج قيمة مسجل المقطع (عنوان المقطع) و قيمة المسجل المساعد له (الإزاحة) نتبع إحدى الطريقتين التاليتين :

الطريقة الأولى

1- نأخذ الخانات الأربعة اليمينية من العنوان الفيزيائي المعطى و نعتبرها إزاحة (أي نضع قيمتها في المسجل المساعد) .
2- نصفر الخانات الأربعة الأولى من العنوان الفيزيائي فينتج معنا رقم ست عشري أول أربع خانات منه أصفاراً .
3- نحذف الصفر الأول من الرقم الناتج فينتج معنا رقم ست عشري هو قيمة مسجل المقطع .

مثال:

بفرض لدينا عدد موجود في العنوان الفيزيائي 41000h أوجد قيمة مسجل المعطيات DS و قيمة المسجل المساعد له SI .

الحل: حسب الطريقة بأخذ الخانات الأربعة الأولى من على اليمين تكون قيمة SI تساوي 1000h و هي الإزاحة.

3-2) DS = 4000h

طريقه أخرى

1- نأخذ الخانة الأولى من العنوان الفيزيائي ونعتبرها إزاحة.

2- نحذف تلك الخانة من العنوان الفيزيائي فيصبح الرقم الناتج مؤلف من أربع خانات و هو يمثل قيمة مسجل المقطع.

مثال: بفرض كان PA = 41000h

الحل : بأخذ الخانة الأولى

1) SI=0

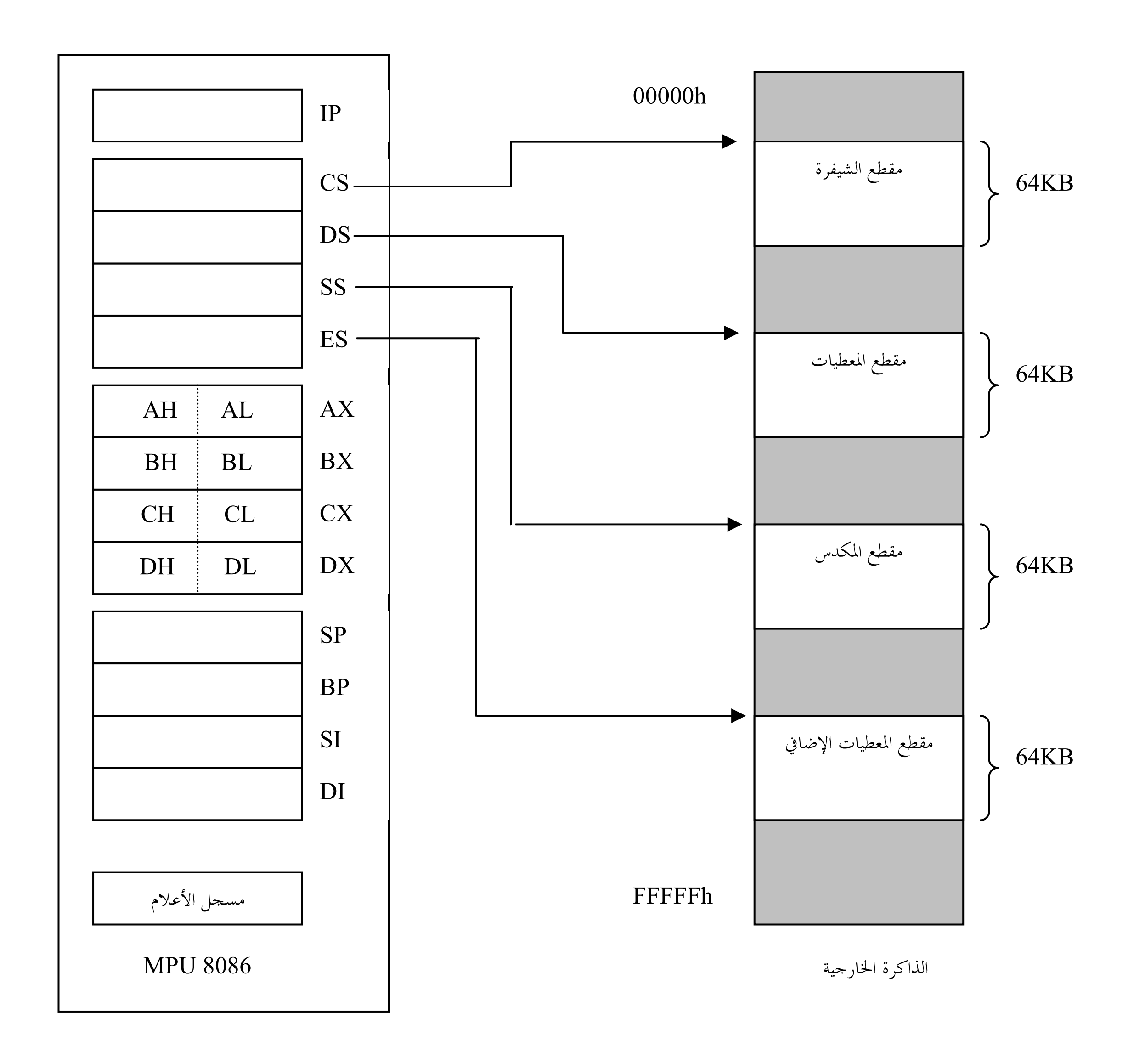
2) DS=4100 أخذنا الخانات المتبقية من الرقم

أي أن

4100:0000 ≡ 4000:1000

إزاحته عنوان ≡ إزاحته عنوان

الموديل البرمجي للمعالج 8086



أساليب العنونة

مقدمة:

إن حيز الذاكرة منظم على شكل بايتات معنونة من 00000h إلى FFFFFh لذلك من أجل كلمات المعطيات 16 بت يتم تخزين البايت السفلي في العنوان الأصغر و البايت العلوي في العنوان الأكبر كما نعلم أن الذاكرة تحتوي أربع مقاطع كل منها 64KB و هي مقطع الشيفرة و مقطع المعطيات و مقطع المكدس و مقطع المعطيات الإضافي، حيث يتم الرجوع إلى هذه المقاطع بمساعدة مسجلات المقاطع ذات الـ16 بت و هي CS, DS, SS, ES و كل من هذه المسجلات يحتوي عنواناً قاعدياً ذا 16 بت و الذي يستخدم في توليد العنوان الفيزيائي للذاكرة و الذي يشير إلى بداية المقطع المطابق في الذاكرة.

يستطيع المبرمج تبديل القيم في مسجلات المقاطع برمجياً، فمثلاً : يمكن تهيئة مقطع معطيات جديد ببساطة و ذلك بتبديل قيمة المسجل DS عن طريق تنفيذ التعليمتين التاليتين :

```
Mov AX,A000
Mov DS,AX
```

و سبب وجود هاتين التعليمتين هو عدم وجود تعليمة واحدة لتحميل مسجل مقطع بعدد ثابت.

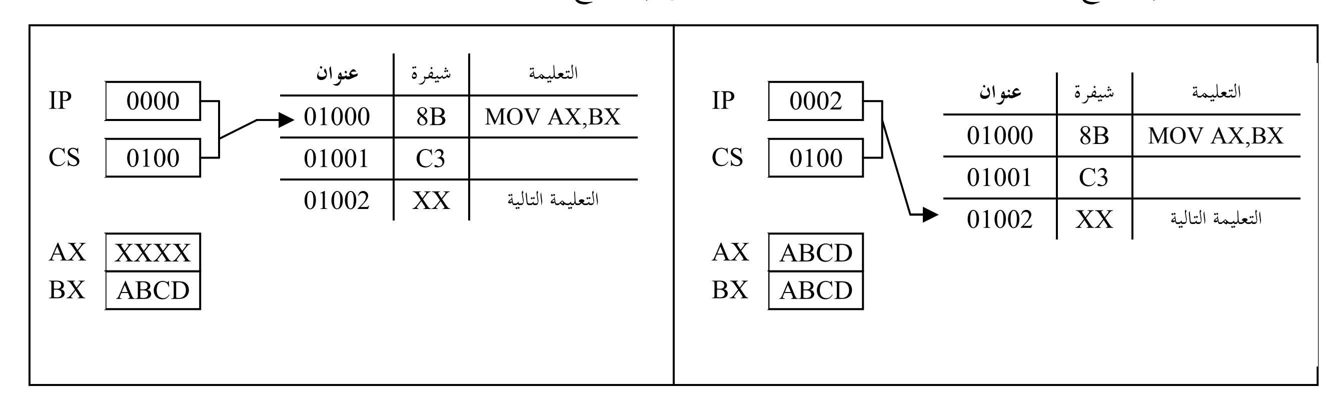
إن المعالج 8086 مزود بتسعة أنظمة عنونة مختلفة، و هي: العنونة بالمسجل – العنونة الفورية – العنونة المباشرة – العنونة غير المباشرة بالمسجل – العنونة القاعدية – العنونة المفهرسة – العنونة القاعدية المفهرسة – العنونة بالسلسلة – العنونة بالنافذة.

و هذه الأنظمة التسعة عدا العنونة بالمسجل و العنونة الفورية تتطلب الرجوع إلى المتحول المخزن في الذاكرة لذلك نحتاج لأن تبدأ وحدة ملائمة الممرات BIU بدورة ممر لقراءة أو كتابة في الذاكرة و هكذا فإن كل نظام عنونة له طريقة مختلفة لحساب عنوان المتحول الذي سيخرج على ممر العناوين أثناء دورة الممر، و سندرس الآن كلاً من هذه الأنظمة بالتفصيل:

ملاحظة: جميع التعليمات ستشرح لاحقاً.

أولاً: نظام العنونة بالمسجل

في هذا النظام يكمن المتحول بمسجل داخلي للمعالج، فمثلاً التعليمة التي تستعمل نظام العنونة بالمسجل هي MOV AX,BX و التي تعني نقل محتوى BX (متحول المصدر) إلى المسجل AX (متحول الهدف) أي أن تنفيذ هذه التعليمة يتم دون الرجوع إلى الذاكرة أي في المعالج لأن كلا المسجلين AX و BX موجودين في المعالج:

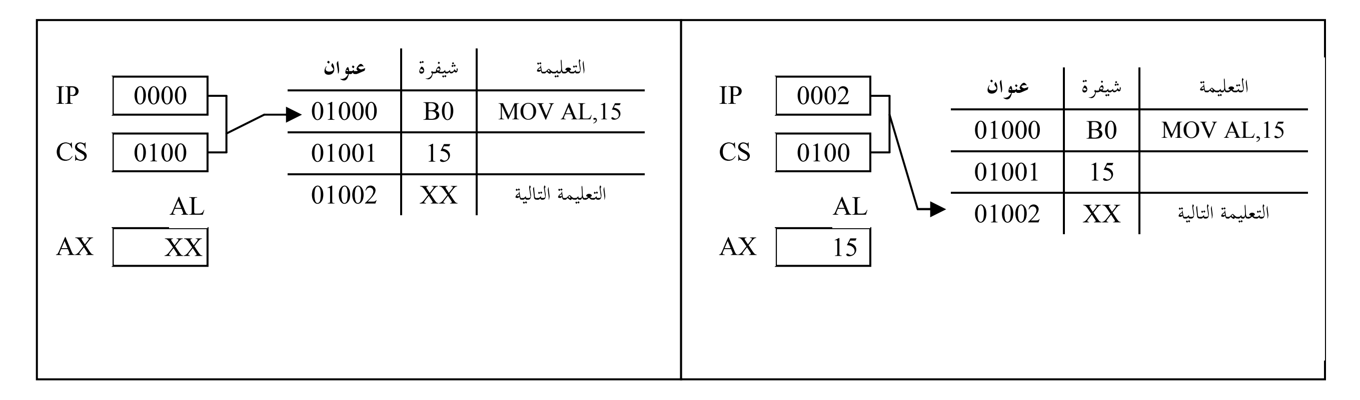


نلاحظ من الشكلين السابقين و في الشكل الأول نجد أنه قد تم توليد العنوان الفيزيائي للتعليمة بواسطة الـــIP و الـــ CS حيث يتم إحضار التعليمة إلى المعالج و تتم فك شيفرتها (8BC3 من الجدول) .

ثانياً: نظام العنونة الفورية

في هذا النظام يكون المتحول جزء من التعليمة و ليس مضمون سجل أو عنوان حجرة ذاكرة حيث يدعى هذا المتحول بالمتحول الفوري و المتحولات الفورية تمثل معطيات ثابتة يمكن أن تكون بايت أو كلمة (2 بايت).

مثال: MOV AL,15 نجد أن متحول المصدر هو 15h و هو متحول مصدر فوري ذو بايت واحد و الشكلان التاليان يوضحان حالة المعالج قبل و بعد تنفيذ التعليمة السابقة.



ثالثاً: نظام العنونة المباشرة

يختلف هذا النظام عن نظام العنونة الفورية بأن الحجرات التي تلي رمز التعليمة تحوي على العنوان الفعال للذاكرة EA = Effective memory Address أي الإزاحة و هذا العنوان مؤلف من 16 بت حيث يتم توليد العنوان الفيزيائي انطلاقاً من DS و ES .

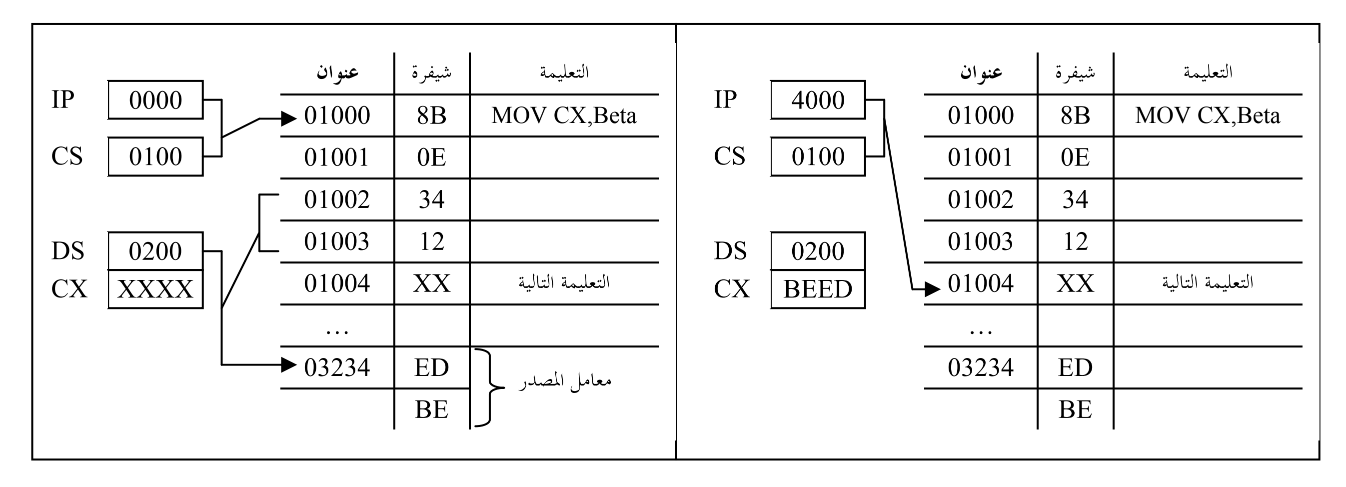
مثال:

MOV CX,[1234]

بفرض كان DS = 200 عندئذ العنوان الفيزيائي يحسب بالعلاقة :

PA = 200 x 10h + 1134 = 03243h

ثم يذهب المعالج إلى الموقع 03234h في الذاكرة و يأخذ محتوى تلك الحجرة و يضعها في CL و يأخذ محتوى الحجرة التي تليها و يضعها في CH .

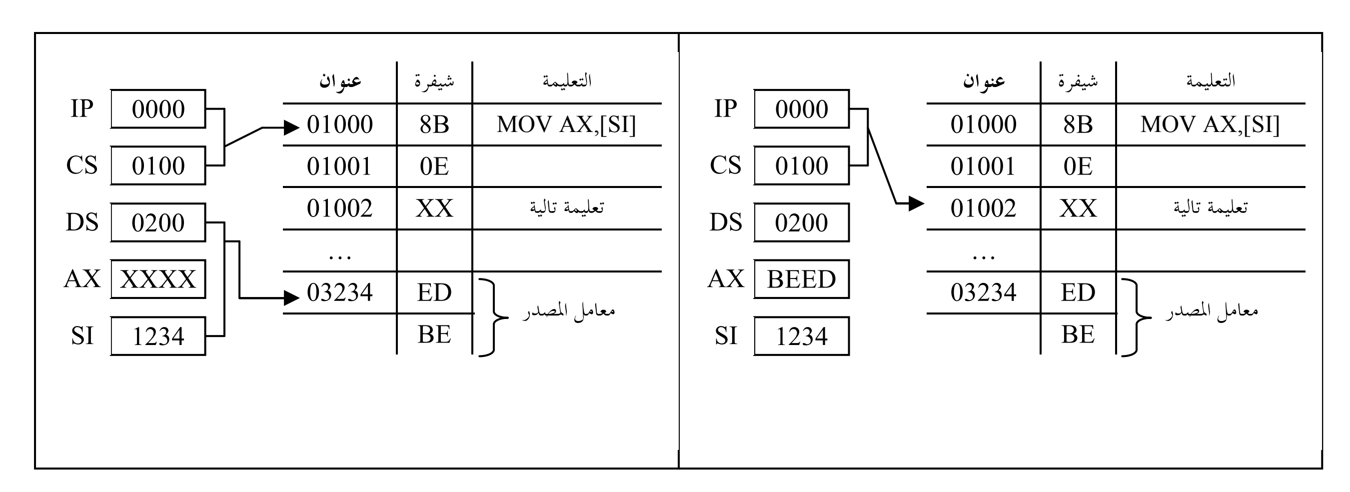


رابعاً: نظام العنونة غير المباشرة بالمسجل :

هذا النظام يشبه نظام العنونة المباشرة لكن يختلف عنه بأن العنوان الفعال (إزاحة) يكمن في مسجل مؤشر BX,BP أو مسجل دليل SI,DI .

مثال:

إن التعليمة التي تستخدم نظام العنونة غير المباشرة بالمسجل هي MOV AX,[SI] حيث يتم توليد العنوان الفيزيائي للمتحول بالاستناد إلى SI و DS . عن طريق العلاقة PA = (DS x 10h) + SI ، و بفرض كانت SI = 1234 و DS = 200 فإن PA = (0200 x 10) + 1234 = 03234 و هو معامل المصدر حيث يذهب المعالج إلى الحجرة 03234 و يأخذ منها قيمتها و يضعها في AL أما قيمة الحجرة التي تليها فيتم وضعها في AH و يبين الشكل التالي حالة المعالج قبل و بعد تنفيذ التعليمة السابقة:



خامساً: نظام العنونة القاعدية

في هذا النظام من العنونة يحسب العنوان بواسطة جمع الإزاحة (disp) مع محتويات إما مسجل القاعدة BX أو مسجل مؤشر القاعدة BP مع القيمة الحالية الموجودة في المسجل DS أو SS على الترتيب أي:

PA = (DS X 10h) + BX + disp = (SS x 10h) + BP + disp

إن تعليمة MOV التي تستخدم العنونة القاعدية لتحديد حجرة متحول الهدف هي

MOV [BX].Beta,AL

إن شيفرة التعليمة السابقة هي 3412 8870 و إن هذه التعليمة تستخدم مسجل القاعدة BX و الإزاحة المباشرة Beta لاشتقاق العنوان الفعال لمتحول الهدف حيث يتم تحقيق نظام العنونة القاعدية بواسطة تخصيص مسجل القاعدة أو مسجل مؤشر القاعدة بقوسين متوسطين (مربعين) متبوعاً بنقطة و إزاحة مباشرة (Beta). إن متحول المصدر في هذه التعليمة متوضع في البايت السفلي من المراكم أي في AL و بفرض أن قيمة Beta هي 1234h فإن العنوان الفيزيائي لمتحول الهدف يتم حسابه بالعلاقة:

PA = (DS x 10h) + BX + disp = 02000 + 1000 + 1234 = 04234h

السجل	القيمة
IP	0000
CS	0100
DS	0200
AX	XX BE
SI	2000

عنوان	شيفرة	التعليمة
01000	8A	MOV AL,array[SI]
01001	44	
01002	34	
01003	12	
01004	XX	التعليمة التالية
…		
02000		
02001		
…		
05234		

السجل	القيمة
IP	0000
CS	0100
DS	0200
AX	XXXX
SI	2000

عنوان	شيفرة	التعليمة
01000	8A	MOV AL,array[SI]
01001	44	
01002	34	
01003	12	
01004	XX	التعليمة التالية
…		
02000	XX	
02001	XX	
…		
05234	BE	معامل المصدر

هذا العنوان الفيزيائي تحسبه الـــ BIU و من ثم تطلب الـــ EU بدء دورة ممر كتابة في الذاكرة و هكذا يكتب متحول المصدر AL في حجرة الذاكرة ذات العنوان الفيزيائي 04234h أي بعد تنفيذ التعليمة تصبح حالة المعالج كما هو واضح في الـــشكل السابق.

سادساً: نظام العنونة المفهرسة

في هذه الطريقة من العنونة يتم الحصول على العنوان الفعال نتيجة جمع محتوى مسجل الفهرس إما DI أو SI إلى عنوان الإزاحة disp (displacement) و هذا النوع من العنونة يناسب أغراض الجداول حيث يكون عنوان الإزاحة في بداية أول عنوان من الجدول و مسجل الفهرس يؤشر إلى أي عنصر من محتويات الجدول.

مثال: ليكن لدينا التعليمة التالية و التي شيفرها 8A443412 و هي MOV AL,array[SI] .

هذه التعليمة يتم فيها تحديد متحول المصدر بواسطة العنونة المفهرسة المباشرة حيث أن array تمثل الإزاحة المباشرة و هي تسبق مسجل الدليل الموجود ضمن قوسين متوسطين، حيث يتم توليد العنوان الفيزيائي التالي:

PA = (DS x 10h) + EA

; EA = (SI) + disp => EA = 2000 + 1234 = 3234h

PA = (DS x 10h) + EA = 02000 + 3234 = 05234h

السجل	القيمة
IP	4000
CS	0100
DS	0200
AX	BE ED
BX	2000

عنوان	شيفرة	التعليمة
01000	8A	MOV [BX].Beta,AL
01001	44	
01002	34	
01003	12	
01004	XX	التعليمة التالية
…		
02000	XX	
02001	XX	

السجل	القيمة
IP	0000
CS	0100
DS	0200
AX	BE ED
BX	1000

عنوان	شيفرة	التعليمة
01000	88	MOV [BX].Beta,AL
01001	07	
01002	34	
01003	12	
01004	XX	التعليمة التالية
…		
02000	XX	
02001	XX	

إن نتيجة تنفيذ هذه التعليمة هي أن محتويات حجرة الذاكرة ذات العنوان الفيزيائي 05234h تنقل إلى AL و تصبح حالة المعالج كما هو موضح في الشكل السابق.

سابعاً: نظام العنونة القاعدية المفهرسة :

في هذا النوع من العنونة يتم الحصول على العنوان الفعال نتيجة جمع محتوى مسجل القاعدة مع مسجل الفهرس و في حال وجود عنوان يجب إضافة هذا العنوان إلى محتوى المسجلين المذكورين.

مثال: التعليمة MOV AH,[BX].Beta[SI] يتم فيها حساب العنوان الفعال لمتحول المصدر كما يلي:

EA = BX + Beta + SI
PA = (DS x 10h) + EA = 02000 + 4234 = 06234h

و يبين الشكل التالي حالة المعالج قبل و بعد تنفيذ التعليمة:

IP	0000
CS	0100
DS	0200
AX	XX XX
BX	1000
SI	2000

عنوان	شيفرة	التعليمة
01000	8A	MOV AH ,[BX].Beta[SI]
01001	20	
01002	34	
01003	12	
01004	XX	التعليمة التالية
…		
02000	XX	
02001	XX	
…		
06234	BE	معامل المصدر

IP	0000
CS	0100
DS	0200
AX	BE XX
BX	1000
SI	2000

عنوان	شيفرة	التعليمة
01000	8A	MOV AH ,[BX].Beta[SI]
01001	20	
01002	34	
01003	12	
01004	XX	التعليمة التالية
…		
02000	XX	
02001	XX	
…		
06234	BE	معامل المصدر

حيث نلاحظ أنه بعد تنفيذ التعليمة تصبح محتويات AH = BEh و التي تمثل محتويات حجرة الذاكرة ذات العنوان الفيزيـــائي 06234h .

ثامناً: نظام العنونة بالسلسلة

إن تعليمات السلسلة في مجموعة تعليمات المعالج 8086 تستعمل أوتوماتيكياً مسجل دليل المصدر و مسجل دليل الهدف لتعيين العناوين الفعالة لمتحولي المصدر و الهدف. فمثلاً تعليمة MOVS هي تعليمة النقل للسلسلة، و هي تستخدم SI و المقطع DS من أجل متحول المصدر و DI و المقطع ES من أجل متحول الهدف. و نلاحظ أنه لا SI و لا DI تظهران في تعليمة السلسلة.

تاسعاً: نظام العنونة بالنافذة

يستعمل هذا النظام مع تعليمات الإدخال و الإخراج لنوافذ I/O . من أجل النوافذ في حيِّز عنونة I/O يستخدم فقط نظام العنونة المباشرة و نظام العنونة غير المباشرة لاستعمال المسجل DX . فمثلاً العنونة المباشرة لنافذة دخل تكون كما في التعليمة التالية:

IN AL,15h ⇔ IN AL,[15h]

تعني هذه التعليمة إدخال معطيات ذات بايت واحد من نافذة الدخل ذات العنوان 15h من حيز عنونة I/O إلى المسجل AL.

مثال آخر عن استعمال العنونة غير المباشرة للنافذة من أجل متحول المصدر هو التعليمة التالية:

IN AL,[DX]

هذا يعني إدخال معطيات ذات بايت واحد من نافذة الدخل التي عنوانها يكون محدد بواسطة مضمون مسجل DX فمثلاً: إذا كان DX = 1234h فإن محتويات النافذة ذات العنوان 1234h يتم تحميلها في المسجل AL.

دروس لغة الأسمبلي التابعة لموقع الفريق العربي للبرمجة
http://www.arabteam2000.com

ملاحظة:
هذه الدروس هي بالأساس مقتبسة من منهاج السنة الثانية قسم هندسة الحاسبات بجامعة حلب

الجزء الثاني
تعليمات المعالج 8086

مقدمة في لغة الأسمبلي

هذه اللغة مزودة لوصف كل من العمليات الأساسية التي يمكن إنجازها بواسطة المعالج المصغر، تُكتب تعليمات هذه اللغة باستعمال الرموز الهجائية أو ما يُدعى ALPHANUMERIC بدلاً من الأصفار و الواحدات في شيفرة الآلة للمعالج. إن الصيغة العامة لكتابة الأمر (التعليمة) في لغة الأسمبلي هي:

تعليق ; تعليمة : لافتة

عادة فإن التعليقات أو الملاحظات التي تصف الأوامر توضع على الطرف الأيمن. و هذا النوع من التوثيق بين التعليمة و التعليق يجعل من السهل على المبرمج كتابة و قراءة و تصحيح الشيفرة. و نقصد بكلمة الشيفرة أن البرنامج مكتوب بلغة الآلة للمعالج و الذي يُعرف بشيفرة الهدف object code أما البرنامج المكتوب بلغة الأسمبلي فيدعى بشيفرة المصدر source code . هذا و إن كل تعليمة في برنامج المصدر تطابق أمراً واحداً في لغة الأسمبلي حيث أن الأمر يجب أن يحدِّد أي عملية سيتم تنفيذها و ما هي متحولات المعطيات التي ستُعالج. لهذا السبب تُقسم التعليمة إلى قسمين منفصلين : رمز التعليمة opcode = operation code و المتحولات operands . رمز العملية هو جزء من التعليمة و الذي يحدد العملية التي ستُنفذ فمثلاً نذكر بعض العمليات النموذجية كالجمع و الطرح و النقل.

في لغة الأسمبلي تستخدم الكلمات المختزلة mnemonic من أجل التعليمات فمثلاً بالنسبة للمعالج 8086 فالكلمات المختزلة في لغة الأسمبلي لعمليات الجمع و الطرح و النقل هي على الترتيب ADD و SUB و MOV . أما المتحولات فتحدد المعطيات التي ستُعالج من قِبل المعالج بواسطة رمز العملية للتعليمة فمثلاً في التعليمة التي تضيف محتويات مسجل القاعدة إلى محتويات المراكم فإن BX و AX هي المتحولات و تُكتب التعليمة على الشكل التالي ADD AX,BX ففي هذا المثال تُضاف محتويات BX إلى AX و يوضع ناتج الجمع في AX و لذلك يُعتبر BX متحول المصدر و AX متحول الهدف.

طاقم تعليمات المعالج 8086

يُزود المعالج 8086 بمجموعة تعليمات مؤلفة من 117 تعليمة أساسية و كذلك إن المجال الواسع للمتحولات و أنظمة العنونة المسموحة للاستعمال مع هذه التعليمات يوسع مجموعة التعليمات إلى تعليمات أكثر، فمثلاً تعليمة Mov الأساسية تمتد إلى 28 تعليمة مختلفة و قابلة للتنفيذ على مستوى لغة الآلة.

أولاً – تعليمات نقل المعطيات

يملك المعالج مجموعة تعليمات وظيفتها نقل المعطيات و ذلك إما بين مسجلات المعالج الداخلية أو بين مسجل داخلي و حجرة تخزين في الذاكرة و هي:

1) تعليمة Mov

تستخدم هذه التعليمة لنقل بايت أو كلمة معطيات من متحول المصدر إلى متحول الهدف و لها الشكل التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	$S \rightarrow D$	MOV D,S	نقل	MOV

إن S,D لهذه التعليمة يمكن أن تكون مسجلات داخلية أو حجرات تخزين في الذاكرة و يبين الجدول التالي مختلف أنواع متحولات المصدر و الهدف مع مثال لكل منها :

الرمز	المعنى
Acc	المراكم
Mem	حجرة ذاكرة
Reg	مسجل
Imm	متحول فوري
Seg-reg	متحول مقطع
Reg16	مسجل ذو 16 بت
Mem16	حجرتي ذاكرة

Source المصدر	Destination الهدف
Acc	Mem
Mem	Acc
Reg	Reg
Mem	Reg
Reg	Mem
Imm	Reg
Imm	Mem
Reg16	Seg-reg
Mem16	Seg-reg
Seg-reg	Reg16
Seg-reg	Mem16

الحالات المستثناة من تعليمة MOV

1- لا تستطيع تعليمة MOV أن تنقل المعطيات بشكل مباشر بين حجرتي ذاكرة لذلك لا نرى في الجدول المجاور الحالة التالية : Mem → Mem و لحل هذه المشكلة فإن المعطيات المرغوب بنقلها يجب نقلها أولاً في مسجل داخلي بواسطة تعليمة MOV ، و من ثم تنقل محتويات هذا المسجل إلى حجرة جديدة في الذاكرة بواسطة تعليمة MOV أخرى.

2- لا يمكن وضع قيمة فورية في مسجل مقطع مباشرة. أي أن التعليمة التالية غير مسموح بها MOV DS,1000 و لحل هذا المشكلة نستخدم التعليمتين التاليتين :

MOV AX,1000

MOV DS,AX

3- لا يمكن نقل محتويات أحد مسجلات المقاطع إلى مسجل مقطع آخر مباشرة، أي أن التعليمة التالية غير مسموح بها MOV DS,ES و لحل هذه المشكلة نقوم بـ

MOV AX,ES

MOV DS,AX

مثال عام : MOV AL,[SI] هذه التعليمة تعني نقل محتويات حجرة الذاكرة المشار إليها بواسطة المسجل SI إلى المسجل AL و إن نظام العنونة في هذه التعليمة هو عنونة غير مباشرة بالمسجل و متحول المصدر عنوانه الفيزيائي هو PA = DS x 10h + SI أما متحول الهدف فهو AL .

2) تعليمة التبديل XCHG

تُستخدم هذه التعليمة لاستبدال متحول المصدر بمتحول الهدف و لاستبدال متحول الهدف بمتحول المصدر.

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	S → D D → S	XCHG D,S	تبديل	XCHG

و يبين الجدول التالي مختلف أنواع متحولات المصدر و الهدف لتعليمة XCHG .

المصدر S	الهدف D
Reg16	Acc
Reg	Mem
Reg	Reg

مثال:

XCHG AX,BX

في هذا المثال يتم التبديل بين محتويات AX و BX.

XCHG [SUM],BX

يتم التبديل بين محتوى الحجرة SUM في الذاكرة و بين المسجل BX .

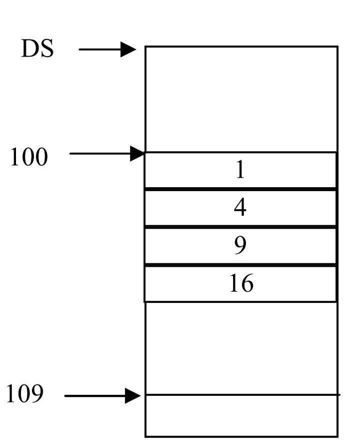
3) تعليمة XLAT

تتعامل هذه التعليمة مع AL فقط ، إن تعامل هذه التعليمة يتم مع الجداول المخزنة في الذاكرة فلو وضعنا في BX إزاحة بداية الجدول نسبة إلى مقطع المعطيات DS و وضعنا في AL إزاحة العنصر نسبه إلى بداية الجدول، عندها تقوم تعليمة XLAT بجمع محتويات المسجل AL مع محتويات المسجل BX و تعتبر الناتج إزاحة بالنسبة إلى مقطع المعطيات، ثم تقوم بوضع قيمة الحجرة المعطى إزاحتها في AL .

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	(DS x 10h) + [AL+BX] → AL	XLAT جدول المصدر	ترجمة	XLAT

مثال:

بفرض أنه لدينا جدول في الذاكرة يحتوي على مربعات الأعداد من 1 إلى 9 أي أنه في أول حجرة من الجدول يوضع مربع العدد 1 و في الحجرة الثانية يوضع مربع العدد 2 (أي 4) ، و هكذا ... ومن هـذا نـرى أن الجدول طوله تسع بايتات إزاحة بدايته عن بداية مقطع الـ DS هي 100 .



عندما يطلب منا الحصول على مربع أحد هذه الأعداد و ليكن العدد 4 أي أن المطلوب هو أن تصبح قيمة AL = 16 لذلك نقوم بما يلي:

1) نضع AL = 3 و BX = 100 .

2) نعطي التعليمة XLAT .

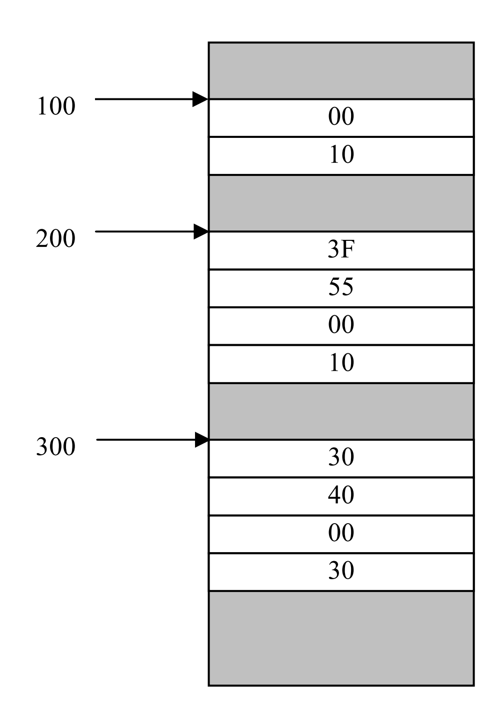
و بعد تنفيذها يصبح AL = 16 و هو المطلوب.

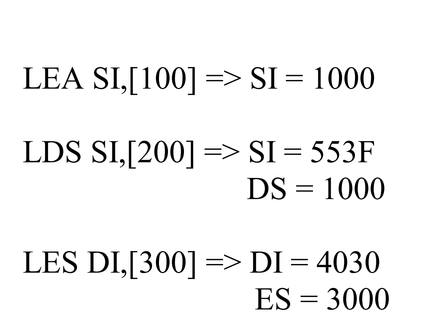
4) التعليمات LEA, LES, LDS

تستعمل هذه التعليمات من أجل عملية نقل المعطيات لتحميل مسجل مقطع أو مسجل أغراض عامة بعنوان بشكل مباشر من الذاكرة. التعليمة LEA وظيفتها هي تحميل مسجل بعنوان فعال أما LDS فهي لتحميل مسجل ما و مسجل مقطع المعطيات DS و تعليمة LES وظيفتها تحميل مسجل ما و مسجل مقطع المعطيات الإضافي ES . وهذه التعليمات موصوفة كما في الجدول التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	Mem16 → reg16	LEA reg16,mem16	تحميل عنوان فعال	LEA
لا يوجد	Mem32 → reg16 Mem32+2 → DS	LDS reg16,mem32	تحميل مسجل و المسجل DS	LDS
لا يوجد	Mem32 → reg16 Mem32+2 → ES	LES reg16,mem32	تحميل مسجل و المسجل ES	LES

أمثلة:





ثانياً – التعليمات الرياضية

و هي تشمل تعليمات من أجل عمليات الجمع، الطرح، الضرب و القسمة.

1) تعليمات الجمع

و هي موصوفة بالجدول التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة	S + D → D Carry → CF	ADD D,S	جمع	ADD

ADC	جمع مع أخذ الانزياح بعين الاعتبار	ADC D,S	S + D + CF → D Carry → CF	أعلام الحالة
INC	الزيادة بمقدار واحد	INC D	D+1 → D	أعلام الحالة
AAA	تصحيح ناتج جمع عددين بشيفرة الآسكي	AAA	سيتم شرحها لاحقاً	AF,CF
DAA	تصحيح ناتج جمع عددين بشيفرة BCD	DAA	سيتم شرحها لاحقاً	كل أعلام الحالة ماعدا OF

تعليمتي ADD و ADC

إن المتحولات المسموحة في تعليمات الجمع ADC ,ADD مبينة في الجدول التالي:

الهدف D	المصدر S
Reg	Reg
Reg	Mem
Mem	Reg
Reg	Imm
Mem	Imm
Acc	Imm

و بالنسبة للمتحولات المسموحة في تعليمة INC فهي :

الهدف D
Reg8
Reg16
Mem

مثال: بفرض AX = 4F3Dh و BX = FD81h و CF = 1 فما هي نتيجة تنفيذ التعليمة ADC AX,BX ؟ مبيناً حالة أعلام الحالة بعد تنفيذ عملية الجمع هذه .

الحل: سنكتب الشيفرة الثنائية للمتحولات من أجل توضيح حالة الأعلام

```
      1111 111          1
AX = 0100 1111 0011 1101 b
BX = 1111 1101 1000 0001 b
CF =                0001 b    +
[1]  0100 1100 1011 1111 b
CF
```

و الآن أعلام الحالة هي:

PF = 0 لأن عدد الواحدات فردي في البايت الأول من ناتج الجمع

AF = 0 لأنه لا يوجد انزياح من الخانة 3 إلى الخانة 4 في البايت الأول من ناتج الجمع (حيث يتم ترقيم الخانات بدءاً من الصفر)

SF = 0 و هي آخر خانة من نتيجة الجمع (الناتج موجب)

CF = 1 بسبب وجود انزياح خارجي

OF = 0 لأنه يوجد إنزياح داخلي و إنزياح خارجي

ملاحظة: الانزياح الداخلي هو الداخل إلى الخانة ذات الأهمية العظمى MSB

ملاحظة: OF = 1 إذا وجد انزياح داخلي فقط أو وجد انزياح خارجي فقط

تعليمة التصحيح DAA

تستخدم هذه التعليمة لإنجاز عملية تصحيح لناتج جمع عددين بشيفرة BCD (هذا و يجب أن يكون ناتج الجمع حتماً في AL أي في النصف السفلي من المراكم AX) و الجدول التالي يبين الحالات الممكنة لجمع عددين بشيفرة BCD :

+	0	1	2	3	4	5	6	7	8	9
0										9
1									9	10
2								9	10	
3							9	10		
4						9	10			
5					9	10				
6				9	10					15
7			9	10					15	16
8		9	10					15	16	
9	9	10					15	16		18

المنطقة الأولى
المنطقة الثانية
المنطقة الثالثة

المنطقة الأولى أرقامها من 0 إلى 9 و فيها تكون نتيجة الجمع صحيحة و لا تحتوي على انزياح و ليست بحاجة إلى تصحيح مثلاً 7+2=9 و هي أرقام واقعة ضمن نطاق المنطقة الأولى.

المنطقة الثانية أرقامها من 10 إلى 15 و فيها تكون نتيجة الجمع غير صحيحة و بحاجة إلى تصحيح بإضافة العدد 6 فنحصل على رقم و حمل إلى العدد الثاني فمثلاً 9+5=E فبإضافة 6 إلى العدد E يكون الناتج 6+E=14 و بذلك تكون النتيجة صحيحة.

المنطقة الثالثة أرقامها من 16 و حتى 18 و فيها تكون نتيجة الجمع غير صحيحة و بحاجة إلى تصحيح و هنا **تتكون** النتيجة من حاصل جمع مع انزياح.

بما أن ناتج الجمع موجود في AL حيث تمثل AL = Bit7 … Bit0

إن قاعدة التصحيح في هذه التعليمة هي :

```
1) if   Bit3 Bit2 Bit1 Bit0   of  AL > 9   or   AF = 1
    then   AL = AL + 6    ,   AF = 1
2) if   AL > 9Fh    or   CF = 1
   then   AL = AL + 60h    ,    CF =1
```

مثال: بفرض أن AL = 28 BCD و BL = 68 BCD

ما هو ناتج تنفيذ ما يلي:

```
ADD AL,BL
DAA
```

الحل: إن نتيجة تنفيذ هاتين التعليمتين هي

```
                 1
28 BCD = 0010   1000 b
68 BCD = 0110   1000 b    +
         1001   0000  →  AL
CF = 0          0110     +
         1001   0110  => AL = 96 BCD
                             AF = 1
```

حيث 0110 تمثل الرقم ستة

تعليمة AAA

تستخدم هذه التعليمة لتصحيح ناتج جمع عددين بشيفرة آسكي (و هنا أيضاً يجب أن يكون ناتج الجمع في المسجل AL) و قاعدة التصحيح في هذه التعليمة هي:

```
if  Bit3 Bit2 Bit1 Bit0   of   AL > 9    or   AF =1
   then   AL = AL + 06
             AL = AL    and    0Fh
             AH = AH + 1
             AF = 1
             CF = 1
   Else   AL = AL     and    0Fh
          AH = 00
```

مثال: بفرض أن AL =32h = 2 ASCII و BL = 34h = 4 ASCII ما هو ناتج تنفيذ التعليمتين التاليتين:

```
ADD AL,BL
AAA
```

الحل: إن ناتج تنفيذ هاتين التعليمتين هو كالتالي :

```
AL = 0011   0010
BL = 0011   0100   +
     0110   0110      →    AL = 66h
AL = 06h , AH = 00
```

و هنا AF = 0 بسبب عدم وجود انزياح من الخانة 3 إلى الخانة 4 (حيث يبدأ الترقيم اعتباراً من الصفر)

2) تعليمات الطرح

هناك مجموعة واسعة من تعليمات الطرح كما هو واضح من الجدول التالي:

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
SUB	طرح	SUB D,S	D - S → D burrow → CF	أعلام الحالة
SBB	الطرح مع الاستعارة	SBB D,S	D - S - CF → D Carry → CF	أعلام الحالة
DEC	الإنقاص بمقدار واحد	DEC D	D-1 → D	أعلام الحالة
NEG	المتمم الثنائي	NEG D	0 – D → D 1 → CF	أعلام الحالة

كل أعلام الحالة عدا OF	سيتم شرحها لاحقاً	DAS	تصحيح ناتج طرح عددين بشيفرة BCD	DAS
AF, CF	سيتم شرحها لاحقاً	AAS	تصحيح ناتج جمع عددين بشيفرة الآسكي	AAS

ملاحظة: إن المتحولات المستخدمة من أجل تعليمتي الطرح SUB, SBB هي نفسها المتحولات المسموحة من أجل تعليمتي الجمع ADD,ADC أما بالنسبة إلى المتحولات المستخدمة من أجل تعليمة DEC فهي نفسها المتحولات المسموحة من أجل تعليمة INC و بالنسبة لتعليمة NEG فالمتحولات المسموحة هي Reg, Reg16, Mem, Mem16 .

مثال: بفرض أن SI = 0018h و DS = 2F00h و العنوان الفيزيائي المتولد عنهما هو 2F018h و بفرض كانت محتويات الحجرة التي يشير إليها العنوان الفيزيائي [2F018] = 0400h ، ما هو ناتج تنفيذ التعليمة SUB [SI],03F8h .

الحل:

تقوم هذه التعليمة بطرح محتويات متحول المصدر (متحول فوري هنا) من محتويات متحول الهدف (محتويات حجرة ذاكرة هنا) حيث أن تعليمة الطرح تتم بإيجاد **المتمم الثنائي** لمتحول المصدر و من ثم جمعه مع متحول الهدف.

```
  Destination = 0400h = 0000 0100 0000 0000 b
       Source = 03F8h = 1111 1100 0000 1000 b   +
---------------------------------------------------
1                 [ ]   0000 0000 0000 1000 b
```

تذكرة بالمتمم الثنائي (و الذي يشار إليه بوضع خطين فوق العدد الذي نريد إيجاد المتمم الثنائي له) :

إذا أردت الحصول على المتمم الثنائي للعدد 03F8h فاعمل ما يلي:

1) تحويل هذا العدد إلى النظام الثنائي فيصبح 0000 0011 1111 1000
2) أقلب الأصفار واحداً و الواحدات أصفاراً فينتج 1111 1100 0000 0111
3) أضف واحد إلى الرقم الناتج فتحصل على المتمم الثنائي 1111 1100 0000 1000 = 03F8h –

لاحظ أن : PF = 0 لأن عدد الواحدات فردي في البايت الأول من الناتج .

AF =1 لأنه لا يوجد معنا حمل (انزياح) عند الانتقال من الخانة الثالثة إلى الخانة الرابعة (عكس حالة الجمع).

ZF = 0 لأن النتيجة ليست صفرية.

SF = 0 و هي قيمة آخر خانة من الناتج MSB.

CF = 0 لأن هناك انزياح خارجي (عكس حالة الجمع).

OF = 0 لحصول انزياح داخلي و انزياح خارجي بآن واحد.

تعليمة DAS

تستخدم هذه التعليمة لتصحيح ناتج طرح عددين بشيفرة BCD حيث يكمن ناتج طرح هذين العددين في المسجل AL و قاعدة التصحيح هي :

1) if Bit3 Bit2 Bit1 Bit0 of AL > 9 or AF = 1

```
   then   AL = AL – 06   ,   AF = 1
2) if   AL > 9Fh   or  CF=1
   then  AL = AL – 60h   ,  CF = 1
```

مثال: بفرض أن AL = 86 BCD و AH = 07 BCD ، بين نتيجة التعليمتين التاليتين:

```
SUB AL,AH
DAS
```

الحل:

```
AL = 1000 0110 b
AH = 1111 1001 b   +
  [1]0111 1111 b   =>  AL = 7Fh
```

و الآن :

AF = 1 بسبب عدم وجود انزياح من الخانة الثالثة إلى الخانة الرابعة.

CF = 0 لوجود انزياح خارجي

و بتطبيق الشرط 1 من قاعدة التصحيح نجد أن AF = 1 , AL = 79h

تعليمة AAS

تستخدم هذه التعليمة لتصحيح ناتج طرح عددين بالشيفرة ASCII حيث يكمن ناتج الطرح في AL ، و قاعدة التصحيح هي:

```
if  Bit3 Bit2 Bit1 Bit0  of  AL > 9   or   AF = 1
   then  AL = AL – 06h   ,   AL = AL  and  0Fh
         AH = AH – 01    ,  AF = 1  ,  CF = 1
   Else  AL = AL   and   0Fh  ,   AH = 00
```

مثال:

بفرض أن AL = 38h = 8 ASCII و BL = 35h = 5 ASCII ، ما هو ناتج تنفيذ التعليمتين التاليتين:

```
SUB AL,BL
AAS
```

الحل:

```
AL = 0011 1000 b
BL = 1100 1011 b    +
  [1]0000 0011 b   =>  AL = 03h
```

AF = 0 بسبب وجود انزياح من الخانة الثالثة إلى الخانة الرابعة

CF = 0 بسبب وجود انزياح خارجي

و بعد تطبيق قاعدة التصحيح نجد AH = 00 , AL = 03h

3) تعليمات الضرب و القسمة

يتم تطبيق هذه التعليمات على الأعداد الثنائية أو بالشيفرة BCD أي في معالجة الأعداد ذات الإشارة و الأعداد بدون إشارة. و هذه التعليمات مبينة في الجدول التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة

أعلام الحالة	AL.S8 → AX AX.S16 → DX,AX	MUL S	ضرب بدن إشارة	MUL
أعلام الحالة	حالة بايت: Q[AX/S8] → AL R[AX/S8] → AH حالة كلمة: Q[(DX,AX)/S16] → AX R[(DX,AX)/S16] → DX	DIV S	تقسيم بدون إشارة	DIV

ببساطة : النقطة تعني عملية الضرب العادية، و الرمز S8 يعني متحول مصدر عبارة عن بايت أما الرمز R فيعني باقي القسمة و الرمز Q ما هو إلا حاصل قسمة.

ملاحظة: إذا كانت قيمة Q في الحالة الأولى (حالة بايت) مساوية لــFF أو كانت قيمة Q في الحالة الثانية (حالة كلمــة) مساوية إلى FFFFh فتحدث مقاطعة من النوع صفر، و تُعرف هذه المقاطعة بخطأ التقسيم.

ملاحظة: بالنسبة لتعليمات الضرب و التقسيم للأعداد ذات الإشارة فهي مشابهة تماماً للتعليمات السابقة و تُعرف كما يلي:

IMUL هي تعليمة الضرب مع أخذ الإشارة بعين الاعتبار.

IDIV هي تعليمة التقسيم مع أخذ الإشارة بعين الاعتبار.

و تكون إشارة الناتج في كلتا التعليمتين آخر خانة منه أي خانة الــ MSB .

و بالإضافة إلى ذلك هناك التعليمات التالية (تابع لجدول الضرب و القسمة):

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة	Q[AL/10d] → AH R[AL/10d] → AL	AAM	تصحيح الناتج في AL من ضرب عددين BCD أو عددين ثنائيين	AAM
SF, ZF, PF	AH.10d + AL → AL **00 → AH**	AAD	تصحيح AX من أجل القسمة حيث AX ليس ناتج القسمة و إنما هو متحول الهدف في عملية القسمة. لذلك نطبق هذه التعليمة قبل تعليمة القسمة على عكس باقي تعليمات التصحيح	AAD
لا يوجد	MSB of AL → **All bits of AH**	CBW	تحويل بايت إلى كلمة	CBW
لا يوجد	MSB of AX → **All bits of DX**	CBW	تحويل كلمة إلى كلمة مضاعفة	CWD

إن المتحولات المسموحة في تعليمات الضرب و القسمة هي بالنسبة للمصدر S :

Mem16, Mem8, Reg16, Reg8 و بالنسبة إلى للهدف D فالمتحول الوحيد المسموح هو المراكم دوماً.

ملاحظة: إن تعليمات القسمة يمكن استخدامها لتقسيم المقسوم بــ 8 بتات في AL على مقسوم عليه بــ 8 بتــات أيــضاً. و لإنجاز هذا يجب أولاً تمديد إشارة المقسوم لملء المسجل AX و هذا يعني ملء AH بأصفار إذا كان العدد موجباً أو بواحدات إذا كان العدد سالباً (أي حسب خانة الإشارة) و تتم هذه العملية بواسطة التعليمة CBW . و بشكل مشابه فإن تعليمات التقسيم

15 0

0		0	

32 بت على 16 بت يمكن استخدامها لتقسيم مقسوم ذي 16 بت في AX على مقسوم عليه ذي 16 بت و ذلك بتحويل الكلمة إلى كلمة مضاعفة و يتم هذا بواسطة التعليمة CWD.

كما ذكرنا سابقاً فإن الأعداد غير المجمعة يتم حفظها كالتالي:

القسم العلوي من البايت الذي يحتوي على العدد غير المجمع يجب أن تكون قيمته مساوية إلى الصفر.

إن التعليمة AAM تستخدم لتصحيح ناتج ضرب عددين غير مجمعين لأنه عند ضرب عددين غير مجمعين نحصل على نتيجة مجمعة و النتيجة يجب أن تكون غير مجمعة، لذلك نصححها بواسطة التعليمة AAM .

مثال: بفرض أن BL = 09 و AL = 07 فما هي نتيجة تنفيذ التعليمات التالية :

```
MUL BL
AAM
```

الحل:

```
  AX = 00 07
  BX = 00 09
-----------------
MUL   00 3F    AX
AAM   06 03    AX
```

قاعدة التصحيح في تعليمة AAD هي :

إن التقسيم بالنسبة إلى الأعداد غير المجمعة يؤدي إلى الحصول على نتائج خاطئة و لذلك يجب تجميع الأعداد قبل قسمتها. و بفرض أن AX = 0604h (و هي أعداد غير مجمعة) فنتيجة تطبيق تعليمة التصحيح AAD (و التي يتم تطبيقها قبل عملية التقسيم) هي:

```
AL = 06 x 10d + 04h = 64d = 40h  }
AH = 00h                         }   => AX = 0040h
```

ثالثاً – التعليمات المنطقية

تنجز عملياتها المنطقية خانة بخانة على متحولاتها. و الجدول التالي يبين التعليمات المنطقية:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة	$S.D \rightarrow D$	AND D,S	AND المنطقي	AND
أعلام الحالة	$S + D \rightarrow D$	OR D,S	OR المنطقي	OR
أعلام الحالة	$S \oplus D \rightarrow D$	XOR D,S	XOR المنطقي	XOR
لا يوجد	$\overline{D} \rightarrow D$	NOT D	NOT المنطقي	NOT

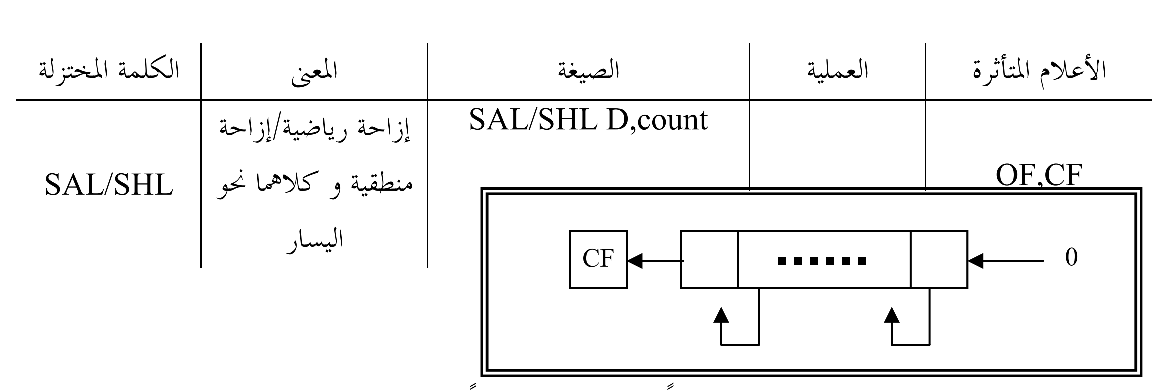
إن المتحولات المسموحة من أجل تعليمة XOR, OR, AND مبينة في الجدول جانباً:

D	S
Reg	Reg
Reg	Mem
Mem	Reg
Reg	Imm
Mem	Imm
AX	Imm

رابعاً – تعليمات الإزاحة

هناك نوعان من تعليمات الإزاحة هما الإزاحة المنطقية و الإزاحة الرياضية كما هو واضح في الجدول التالي:

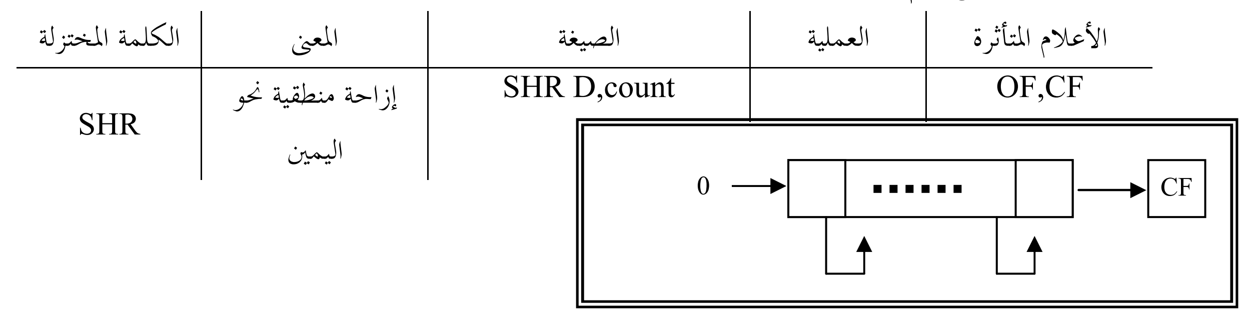
الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزنة
OF,CF		SAL/SHL D,count	إزاحة رياضية/إزاحة منطقية و كلاهما نحو اليسار	SAL/SHL



العملية هنا هي إزاحة محتويات D نحو اليسار باتجاه CF عدداً من الخانات مساوياً لقيمة count و ملء جميع الخانات اليـــمنى المفرغة بأصفار.

و بالنسبة لتأثير هذه التعليمة على علم OF : إذا تبدلت خانة الإشارة نتيجة الإزاحة فإن OF = 1 .

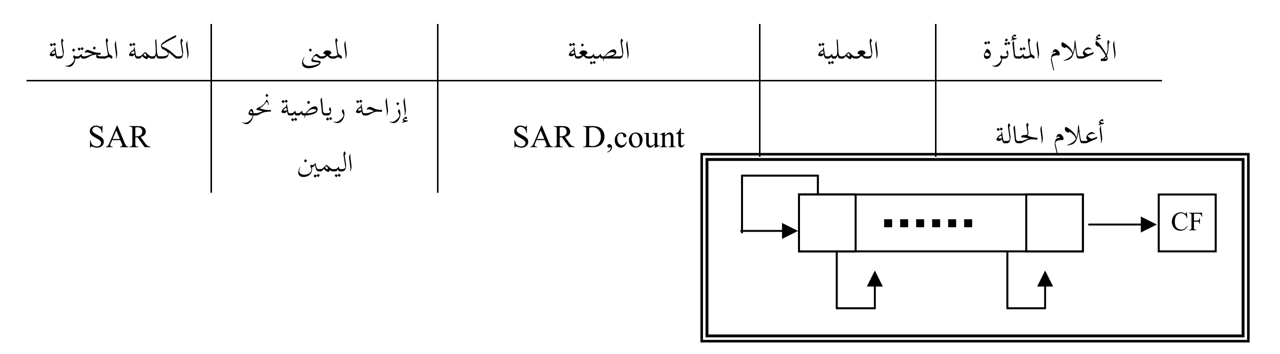
الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزنة
OF,CF		SHR D,count	إزاحة منطقية نحو اليمين	SHR



العملية هنا هي إزاحة محتويات D نحو اليمين باتجاه CF عدداً من الخانات مساوياً لقيمة count و ملء جميع الخانات اليـــسرى المفرغة بأصفار.

و بالنسبة لتأثير هذه التعليمة على العلم OF : إذا تبدلت خانة الإشارة نتيجة الإزاحة فإن OF = 1

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزنة
أعلام الحالة		SAR D,count	إزاحة رياضية نحو اليمين	SAR



العملية هنا هي إزاحة محتويات D نحو اليمين باتجاه CF عدداً من المرات مساوياً لقيمة count و ملء الخانات جميع الخانـــات اليسرى بقيمة الخانة MSB (خانة الإشارة أو آخر خانة).

ملاحظة: بالنسبة للتعليمتين SHL, SAL : إذا طبقنا هاتين التعليمتين من أجل الإزاحة بعدد من الخانات N = count فهـذا يعني ضرب متحول الهدف بـ 2^n و الذي هو مضاعفات العدد 2 .

ملاحظة: إن التعليمة SHR تعني تقسيم متحول الهدف على العدد 2^{count} تحت كون LSB = 0 كل مـرة و في حـالة LSB = 1 فعندها يكون لدينا باقي موضوع في العلم CF .

مثال: اكتب برنامجاً يقوم بحساب العلاقة الرياضية التالية مستخدماً تعليمات الإزاحة و التعليمات الرياضية :

$3.(AX) + 7.(BX) \rightarrow DX$

```
MOV SI,AX        ; copy AX into SI
SAL SI,1         ; 2 AX
ADD SI,AX        ; 3 AX
MOV DX,BX        ; copy BX into DX
MOV CL,03H       ; load shift count
SAL DX,CL        ; 8 BX
SUB DX,BX        ; 7 BX
ADD DX,SI        ; result
```

إن المتحولات المسموحة بالنسبة لتعليمات الإزاحة هي:

D	Count
Reg	1
Reg	CL
Mem	1
Mem	CL

أي عندما Count لا يساوي الواحد فعندئذ يجب تحميل قيمـة count في المسجل CL ثم كتابة تعليمات الإزاحة أي: عندما count يساوي الواحـد فيمكن أن نكتب :

```
SAL AX,1
```

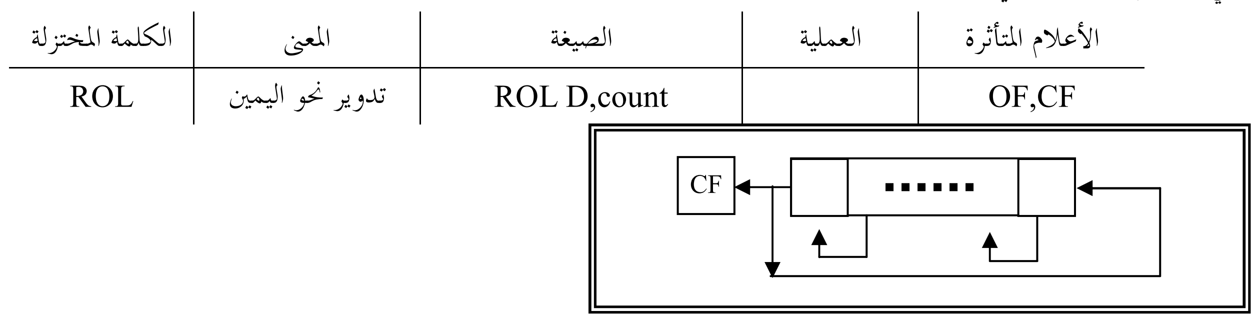
و عندما $count \neq 1$ <= يجب أن نكتب :

```
MOV CL,count
SAL AX,CL
```

هذا و إن قيمة count محددة بالمجال [1,FF] و الأقواس المحيطية ليس لها علاقة بمفهوم الإزاحة طبعاً.

خامساً – تعليمات التدوير

و هي مبينة في الجدول التالي:

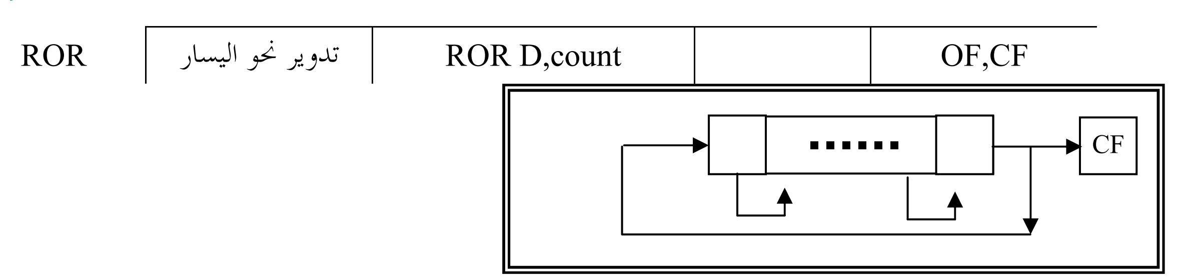
الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
ROL	تدوير نحو اليمين	ROL D,count		OF,CF

العملية هنا هي تدوير محتويات D نحو اليسار عدداً من المرات مساوياً لقيمة count . و كل خانة تُزاح خارج الـــ MSB توضع في الخانة LSB و في CF .

و بالنسبة لتأثير هذه التعليمة على العلم OF فهو نفس المناقشة في التعليمات السابقة.

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة

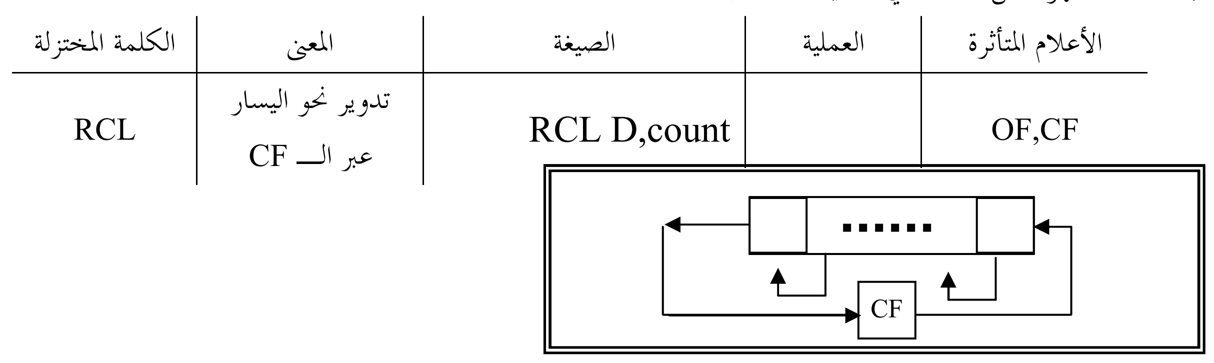
OF,CF		ROR D,count	تدوير نحو اليسار	ROR



العملية هنا هي تدوير محتويات D نحو اليمين عدداً من المرات مساوياً لقيمة count . و كل خانة تُزاح خارج الـــ LSB توضع في الخانة MSB و في CF .

و بالنسبة لــ OF فهو نفس المناقشة في التعليمات السابقة.

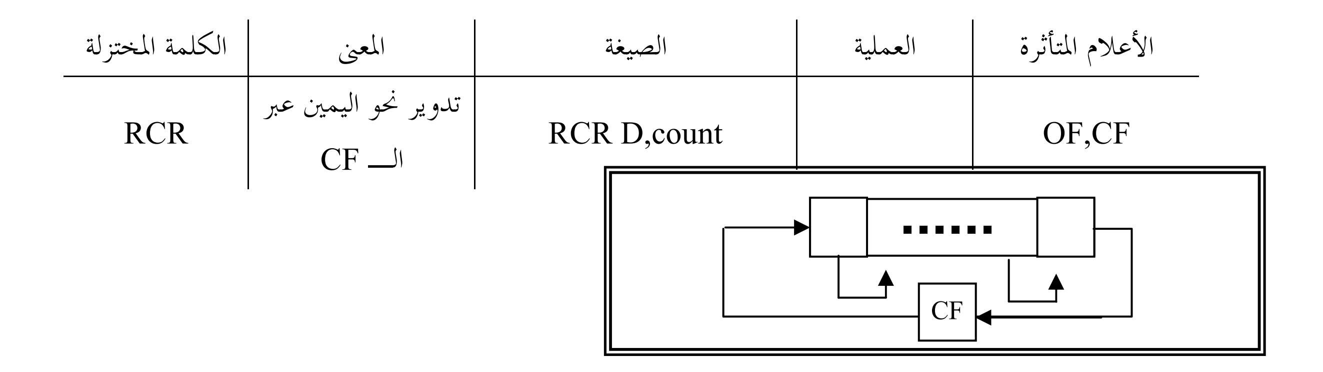
الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
OF,CF		RCL D,count	تدوير نحو اليسار عبر الـــ CF	RCL



العملية هنا مشابهة لتعليمة ROL ما عدا أن المحتوى الأصلي لــ CF يوضع في الخانة LSB أما الخانة المزاحة خارج الـــ MSB فتوضع في CF .

و بالنسبة لــ OF نفس المناقشة السابقة.

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
OF,CF		RCR D,count	تدوير نحو اليمين عبر الـــ CF	RCR



العملية هنا مشابهة لتعليمة ROR ما عدا أن المحتوى الأصلي لــ CF يوضع في الخانة MSB أما الخانة المزاحة خارج الـــ LSB فتوضع في CF .

و بالنسبة لــ OF نفس المناقشة السابقة.

سادساً – تعليمات مسجلات الأعلام

و هي مبينة في الجدول التالي :

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة

LAHF	تحميل AH من مسجل الأعلام	LAHF	Flags → AH النصف الأول من مسجل الأعلام يوضع في AH	لا يوجد
SAHF	تخزين قيمة AH في مسجل الأعلام	SAHF	AH → Flags يوضع AH في النصف الأول من مسجل الأعلام	أعلام الحالة عدا OF
CLC	تنظيف الـ CF	CLC	0 → CF	CF
STC	توضيع الـ CF	STC	1 → CF	CF
CMC	متمم أحادي لـ CF	CMC	$\overline{CF}$ → CF	CF
CLI	تنظيف IF	CLI	0 → IF	IF
STI	توضيع الـ IF	STI	1 → IF	IF

سابعاً – تعليمات المقارنة

تسمح تعليمة المقارنة CMP بمقارنة عددين بـ 8 بت أو 16 بت و هي مشروحة بالجدول التالي:

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
CMP	مقارنة عددين	CMP D,S	D – S تتأثر الأعلام	أعلام الحالة

تجري عملية الطرح ضمنياً دون تخزين نتيجتها في متحول الهدف D (أي تبقى كلاً من محتويات المصدر S و محتويات الهدف D على حالها) و تستعمل هذه التعليمة لجعل أعلام الحالة تأخذ قيمة واحد منطقي أو صفر منطقي. إن المتحولات المسموحة لهذه التعليمة مبينة في الجدول التالي:

D	S
Reg	Reg
Reg	Mem
Mem	Reg
Reg	Imm
Mem	Imm
Acc	Imm

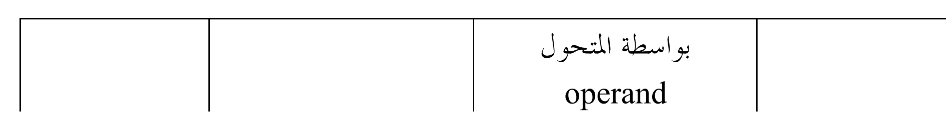
ثامناً – تعليمات القفز

الغاية من تعليمة القفز هي تعديل طريق تنفيذ التعليمات في البرنامج. و هنـــاك نوعان من تعليمات القفز، وهي : القفز المشروط و القفز غير المشروط. في القفز غير المشروط لا يوجد أي شروط مـــن أجـــل حدوث القفز أما في القفز المشروط فإن الحالات الشرطية الموجودة في لحظة تنفيذ تعليمة القفز تتخذ القرار فيما إذا ســـيحدث القفز أم لا، ففي حال تحقق الحالات الشرطية فإنه يتم القفز، و إلا يُتابع التنفيذ بالتعليمة التي تلي تعليمة القفز في البرنامج.

1) تعليمة القفز غير المشروط

و هي مشروحة في الجدول التالي:

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
JMP	قفز غير مشروط	JMP operand	القفز إلى العنوان المحدد	لا يوجد



	بواسطة المتحول operand		

هناك نوعان أساسيان من القفز غير المشروط الأول يدعى بالقفز ضمن المقطع الجزئي، و الثاني هو القفز بين المقاطع الجزئية أي يُمَكِّننا من القفز من أحد مقاطع الشيفرة إلى مقطع آخر و إن تحقيق هذا النوع من القفز يتطلب منا تعديل محتويات كل من مقطع الـCS و مسجل مؤشر التعليمة IP ، أما القفز ضمن المقطع الجزئي فإنه يتطلب منا تعديل قيمة الـIP فقط.

إن المتحولات المسموحة لتعليمة القفز غير المشروط هي :

	Operand	
متحول اللافتة القصيرة	Short_Label	للقفز ضمن المقطع الجزئي
متحول اللافتة القريبة	Near_Label	
متحول مؤشر ذاكري 16 بت	Memptr16	
متحول مؤشر مسجلي 16 بت	Regptr16	
متحول اللافتة البعيدة	Far_Labrl	للقفز بين المقاطع الجزئية
متحول مؤشر ذاكري 32 بت	Memptr32	

القفز ضمن المقطع الجزئي

أ) إن متحولات اللافتة القصيرة و اللافتة القريبة تحدد القفز النسبي لعنوان تعليمة القفز نفسها فمثلاً في تعليمة القفز باللافتة القصيرة يتم تشفير العدد ذي 8 بت كمتحول فوري لتحديد الإزاحة (Disp) ذات الإشارة التي تشير إلى التعليمة التالية التي سيتم تنفيذها من حجرة تعليمة القفز، و عندما تنفذ تعليمة القفز يعاد شحن الـIP بقيمة جديدة موضحة كما يلي:

قيمة IP الجديدة = [(قيمة IP + طول شيفرة تعليمة القفز) + مقدار الإزاحة ذات الإشارة بعد تمديدها بجعل متحول 8 بتات بالشكل 16 بت]

إن القيمة الجديدة لــ IP مع قيمة CS الحالية تعطي العنوان الفيزيائي للتعليمة التالية التي ستجلب و تنفذ.

مثال:

ليكن لدينا

IP = 0112h
JMP disp ; disp = 0F2h

إن عنوان تعليمة القفز (موجود تحت العنوان المخزن في IP) ، إذن سيتم القفز إلى التعليمة ذات العنوان التالي:

address العنوان المنطقي = IP + 2 + disp (بعد تمديد إشارتها) = 0112 + 2 + FFF2 = 0106h (أهملنا خانة الحمل)

بما أن العنوان الناتج أصغر من عنوان تعليمة القفز فهذا يعني أننا نقفز إلى تعليمة تسبق تعليمة القفز أي القفز نحو الوراء . 0106 < 0112 .

مثال آخر:

IP = 0112h
JMP 04
Address = 0112 + 2 + 0004 = 0118h

نلاحظ أن 0112 < 0118 فهذا يعني أن القفز نحو الأمام.

و للحصول على العنوان الفيزيائي يجب إضافة مقدار الـCS لقيمة address .

ملاحظة: بما أن متحول اللافتة القصيرة ذو 8 بتات فهو يسمح بالقفز في المجال من 126- إلى 129+ و سبب ذلك أنه إذا أضفنا طول شيفرة تعليمة القفز و هو 2 بايت إلى المجال التالي من 128- إلى 127+ سنحصل على المجال السابق. أما متحول

اللافتة القريبة فهو متحول فوري ذو 16 بت و لذلك يسمح بالقفز ضمن مجال يساوي 32KB نحو الخلف أو نحو الأمام من عنوان تعليمة القفز.

مثال:

JMP label

هذا يعني القفز إلى نقطة في البرنامج مقابلة للمتحول label حيث تتم إضافة هذا المتحول (الإزاحة 16 بت) إلى قيمة الIP و القيمة الجديدة لـــ IP و القيمة الحالية في CS تعطي العنوان الفيزيائي للتعليمة التي ستنفذ .

ب) يمكن تحديد القفز إلى عنوان بشكل غير مباشر بواسطة محتويات حجرة ذاكرة أو محتويات مسجل أي باستخدام متحول مؤشر ذاكري 16 بت أو متحول مؤشر مسجلي 16 بت و هنا أيضاً يتم القفز ضمن مجال $\pm 32KB$.

مثال:

JMP BX

في هذه التعليمة يُستعمل مضمون المسجل BX من أجل الإزاحة و هذا يعني أن قيمة BX يتم تحميلها في IP ثم يحسب العنوان الفيزيائي للتعليمة التي سيتم القفز إليها باستعمال المحتويات الحالية لـــ CS و القيمة الجديدة لـــ IP .

بفرض أن :

BX = 0200h
CS = 0100h
} العنوان الفيزيائي للتعليمة التي سيتم القفز إليها
PA = (CS x 10h) + BX = 01000 + 0200 = 01200h

ملاحظة : يمكن استخدام مختلف أنواع أنظمة العنونة لتحديد المتحول المستعمل كمؤشر ذاكري فمثلاً JMP [SI] ففي هذه التعليمة تستعمل محتويات SI كعنوان حجرة الذاكرة التي تحتوي على العنوان الفعال، هذا العنوان يتم تحميله في IP و الذي يُستعمل مع محتويات CS الحالية لحساب العنوان الفيزيائي للتعليمة التي سيتم القفز إليها و عادة في هذه الحالة تستخدم المسجلات التالية: DI, SI, BX .

القفز بين المقاطع الجزئية أو القفز خارج المقطع الجزئية

أ) تَستعمِل اللافتة البعيدة متحولاً فورياً ذا 32 بت لتحديد القفز إلى عنوان ما. حيث يتم تحميل الـــ 16 بت الأولى من هذا المتحول في IP و تكون هي العنوان الفعال نسبة لمحتويات المسجل CS أما الـــ 16 بت الثانية فيتم تحميلها في المسجل CS و التي تحدد مقطع الشيفرة الجديد.

مثال:

JMP farlabel

حيث farlabel هو متحول بـــ 32 بت (الكلمة الأول تشحن في IP و الكلمة الثانية تشحن في الـــ CS).

ب) إن الطريقة غير المباشرة لتحديد العنوان الفعال و عنوان مقطع الشيفرة من أجل القفز بين المقاطع الجزئية هي باستعمال متحول مؤشر ذاكري بـــ 32 بت. و في هذه الحالة فإن أربع بايتات من الذاكرة متتابعة اعتباراً من العنوان المحدد تحتوي على العنوان الفعال و عنوان مقطع الشيفرة الجديد على الترتيب. و هنا أيضاً يمكن استخدام أي نوع من أنواع أنظمة العنونة المختلفة،

مثال:

JMP farseg [DI] ففي هذه التعليمة تُستعمل محتويات DS, DI لحساب عنوان حجرة الذاكرة التي تتضمن الكلمة الأولى للمؤشر الذي يُعرِّف الحجرة التي سيتم القفز إليها، فإذا كان :

DI = 0200h
DS = 0100h
} إن العنوان الفيزيائي للمؤشر هو :
PA = DS x 10h + DI = 01000 + 0200 = 01200h

و لتكن محتويات هذه الحجرة و الحجرات التي تليها كما هو واضح في الشكل التالي:

Address (h)	Content
01200	10
01201	30
01202	00
01203	04

قيمة IP الجديدة هي IP = 3010h

قيمة CS الجديدة هي CS = 0400h

إذن العنوان الفيزيائي للتعليمة التي سيتم القفز إليها هو:

PA = CS x 10h + IP = 07010h

2) تعليمة القفز المشروط

و هي مشروحة في الجدول التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	إذا تحقق الشرط cc فإنه يتم القفز إلى العنوان المحدد بواسطة المتحول و إلا فيتم تنفيذ التعليمة التالية لتعليمة القفز	Jcc متحول	قفز مشروط	Jcc

هناك 18 من تعليمات القفز المشروط و هي مشروحة في الجدول التالي:

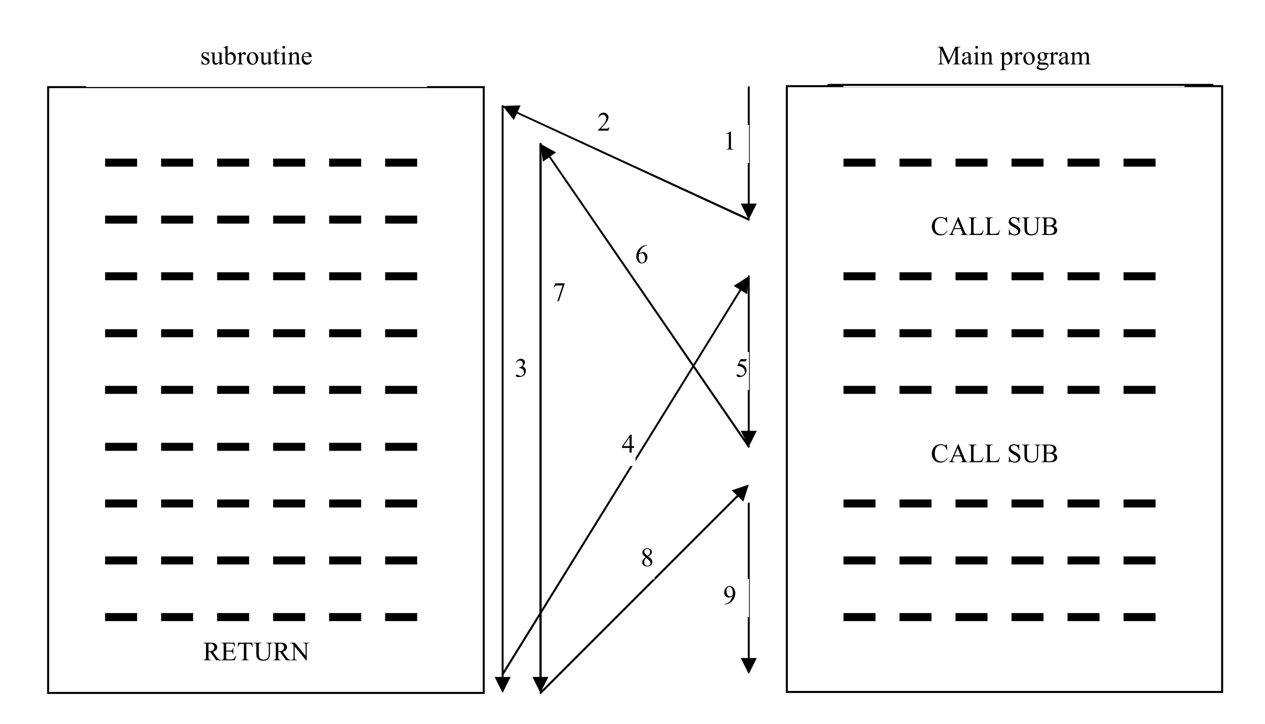
المعنى	الكلمة المختزلة
القفز إذا كان CF = 1	JC
القفز إذا كان CF = 0	JNC
القفز إذا كان OF = 1	JO
القفز إذا كان OF = 0	JNO
القفز إذا كان SF = 1	JS
القفز إذا كان SF = 0	JNS
القفز إذا كان CX = 0000	JCXZ
القفز في حالة التساوي/أو إذا كان الناتج يساوي الصفر	JE/JZ
القفز إذا كان أكبر أو يساوي/القفز إذا لم يكن أصغر	JGE/JNL
القفز إذا كان فوق/القفز إذا لم يكن تحت أو يساوي	JA/JNBE
القفز إذا كان فوق أو يساوي/القفز إذا لم يكن تحت	JAE/JNB
القفز إذا كان تحت/القفز إذا لم يكن فوق أو يساوي	JB/JNAE
القفز إذا كان تحت أو يساوي/القفز إذا لم يكن فوق	JBE/JNA
القفز إذا كان أكبر/القفز إذا لم يكن أصغر أو يساوي	JG/JNLE
القفز إذا كان أصغر أو يساوي/القفز إذا لم يكن أكبر	JLE/JNG
القفز إذا لم يكن يساوي/القفز إذا كان الناتج يساوي قيمة غير صفرية	JNE/JNZ
القفز إذا كانت خانة Parity غير موجودة/القفز إذا كان PF = 0	JNB/JBO
القفز في حالة وجود خانة Parity/القفز إذا كان PF = 1	JP/JPE

ملاحظة:

للتمييز بين مقارنة الأعداد ذات الإشارة و الأعداد بدون إشارة فإن هناك اسمين مختلفين يبدو أنهما نفس الشيء في تعليمات القفز و هما فوق (A) و تحت (B) من أجل مقارنة الأعداد بدون إشارة، و أصغر (L) و أكبر (G) من أجل مقارنة الأعداد ذات الإشارة. فمثلاً العدد ABCDh هو فوق العدد 1234h إذا اعتبرناهما عددين بدون إشارة. أما إذا اعتبرناهما بإشارة فــإن ABCDh هو عدد سالب و 1234h هو عدد موجب و لذلك ABCDh هو أصغر من 1234h.

البرامج الفرعية *SUBROUTINES*

هي إجراءات مكتوبة بشكل مستقل عن البرنامج الرئيسي. متى وجب على البرنامج الرئيسي أن ينجز الوظيفة المحددة بواسطة البرنامج الفرعي فإنه يستدعي البرنامج الفرعي إلى العمل و من أجل هذا يجب أن يتحول التحكم من البرنامج الرئيسي إلى نقطة البداية في البرنامج الفرعي، حيث يستمر تنفيذ البرنامج الفرعي، و عند اكتمال التنفيذ يعود التحكم إلى البرنامج الرئيسي بالتعليمة التالية لتعليمة مناداة البرنامج الفرعي:



ملاحظة:

إن الفرق بين العمل لمناداة البرنامج الفرعي و القفز هو أن مناداة البرنامج الفرعي لا تنتج قفزاً فقط إلى العنوان المناسب في ذاكرة تخزين البرنامج و لكنها أيضاً تملك تقنية من أجل حفظ المعلومات مثل IP و CS التي تكون مطلوبة للعودة إلى البرنامج الرئيسي.

تعليمات المناداة و العودة

كلاً هاتين التعليمتين معاً تُزودان تقنية من أجل استدعاء البرنامج الفرعي إلى العمل و إعادة التحكم إلى البرنامج الأساسي لمتابعة تنفيذه. إن تعليمة المناداة مشروحة في الجدول:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	يُتابع التنفيذ في البرنامج الفرعي من العنوان المحدد بواسطة المتحول operand الموجود في تعليمة المناداة. و المعلومات المطلوبة من أجل العودة مثل IP و CS تُحفظ في المكدس	CALL operand	مناداة برنامج فرعي	CALL

هناك 5 أنواع للمتحولات المسموح باستخدامها مع تعليمة المناداة و هي:

OPERAND	
Near_pro	للمناداة ضمن المقطع الجزئي
Memptr16	
Regptr16	
Far_proc	للمناداة خارج المقطع الجزئي
Memptr32	

إن المتحولات الثلاثة الأولى مخصصة للمناداة ضمن المقطع الجزئي للبرنامج الفرعي (أي البرنامج الرئيسي و البرنامج الفرعي يقعان في نفس مقطع الشيفرة) حيث أن تنفيذ تعليمة المناداة يسبب حفظ محتويات IP في المكدس لأنه سوف يتم تعديل قيمة IP آلياً لتتلائم مع البرنامج الفرعي. و عندئذ ينقص مؤشر المكدس بمقدار 2 ، إن القيمة المحفوظة في IP ضمن المكدس هي عنوان التعليمة التي تلي تعليمة المناداة.

بعد وضع قيمة IP في المكدس (أي حفظ العنوان الذي سنعود إليه بعد تنفيذ البرنامج الفرعي) يتم شحن IP بعنوان و بقيمة جديدة ذات 16 بت هذه القيمة تشير إلى عنوان التعليمة الأولى من تعليمات البرنامج الفرعي المخزنة في الذاكرة، و يمكن ذكر تعليمة المناداة ضمن المقطع الجزئي على الشكل التالي كأمثلة على متحوات الجدول السابق و على الترتيب:

```
CALL near_proc
CALL [SI]
CALL BX
```

أما النوع الآخر لتعليمة المناداة (المناداة خارج المقطع الجزئي) فهو يسمح للبرنامج الفرعي بأن يكمن في مقطع شيفرة آخر، و في هذه الحالة تستخدم المتحولات التالية Far_pro ، Memptr32 كما هو واضح في الجدول السابق. تحدد هذه المتحولات كلاً من العنوان الجديد لـ IP و عنوان المقطع الجديد لـ CS . في كلتا الحالتين فإن تنفيذ تعليمة المناداة يسبب حفظ محتويات المسجلات CS ثم IP في المكدس و من ثم تحميل القيم الجديدة المحددة بالمتحول operand في IP و CS . إن القيم المختزنة لـ CS و IP في المكدس تسمح بالعودة إلى البرنامج الرئيسي من مقطع شيفرة آخر. إن المتحول Far_proc يمثل متحولاً فورياً بـ 32 بت و الذي يكون مخزناً في البايتات الأربعة التي تلي رمز التعليمة (opcode) لتعليمة المناداة في ذاكرة البرنامج.

مثال :

CALL 01234321 حيث أن هاتان الكلمتان يتم تحميلهما مباشرة من ذاكرة تخزين البرنامج في IP و CS حيث CS هو مقطع الشيفرة للبرنامج الفرعي. إن عنوان التعليمة الأولى في البرنامج الفرعي يكون محدداً بالكلمة الأولى بعد تعليمة CALL أي يخزن ضمن IP . أما بالنسبة لمتحول المؤشر من نوع ذاكري بـ 32 بت فإن المؤشر للبرنامج الفرعي يكون مخزناً كأربعة بايتات في ذاكرة المعطيات، و الحجرة الأولى للمؤشر يمكن تحديدها بشكل مباشر بواسطة أحد المسجلات (المثال هنا هو نفس مثال القفز JMP farseg [DI] السابق).

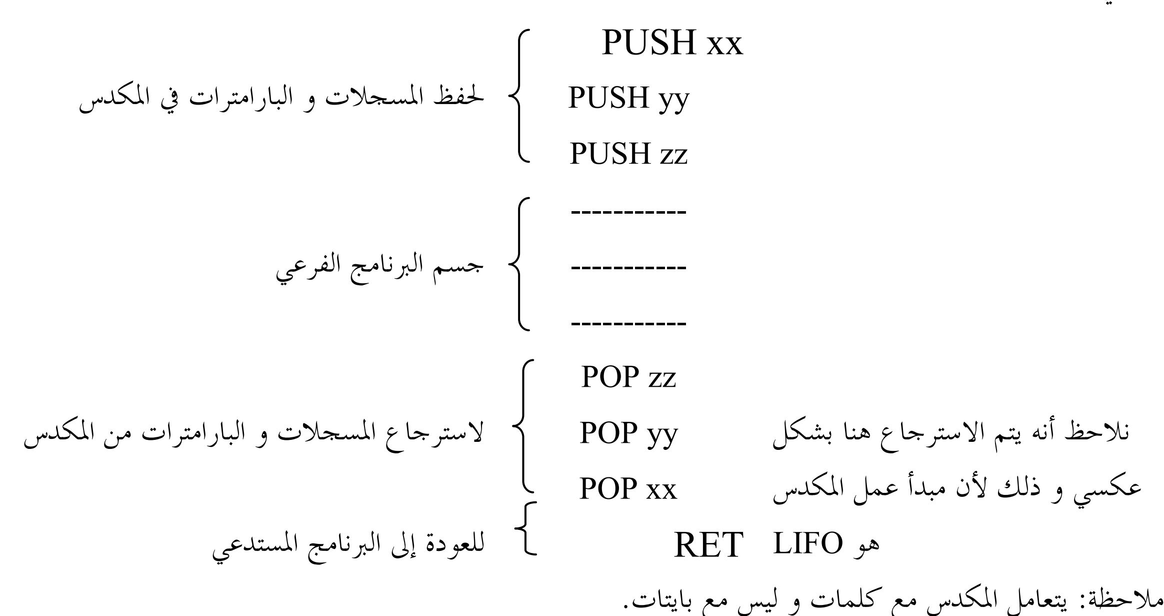
إن كل برنامج فرعي يجب أن ينتهي بتنفيذ التعليمة التي تعيد التحكم إلى البرنامج الرئيسي و هذه التعليمة هي تعليمة العودة RET و هي مشروحة بالجدول التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		RET/RET operand	العودة إلى البرنامج المُستدعِي	RET

العودة إلى البرنامج المستدعي عن طريق إعادة تخزين قيم IP فقط أو IP و CS معاً (حسب نوع تعليمة المناداة أي ضمن المقطع الجزئي أو خارجه) من أجل المتحول Far_pro . و إذا كان المتحول (operand) موجوداً في تعليمة العودة RET فيجب إضافته إلى محتويات SP . هذا و إن المتحول إذا وجد في تعليمة العودة فهو عبارة عن متحول إزاحة بـ 16 بت.

تاسعاً – تعليمات الدفع و السحب

إن التعليمة المستخدمة لحفظ البارامترات في المكدس هي تعليمة الدفع PUSH و التعليمة المستخدمة لاسترجاعها هي تعليمة POP . بعد سياق التحويل إلى البرنامج الفرعي نجد أنه من الضروري عادة حفظ محتويات المسجلات الرئيسية أو بعض بارامترات البرنامج الرئيسي هذه القيم يتم حفظها بواسطة دفعها إلى المكدس. و بهذه الطريقة يتم حفظ المحتويات سليمة في مقطع المكدس للذاكرة أثناء تنفيذ البرنامج الفرعي، و قبل العودة إلى البرنامج الرئيسي فإن المسجلات المحفوظة و بارامترات البرنامج الرئيسي يُعاد تخزينها بواسطة سحب القيم المحفوظة من المكدس. لذلك فإن البنية النموذجية للبرنامج الفرعي تكون كالتالي:



ملاحظة: يتعامل المكدس مع كلمات و ليس مع بايتات.

تعليمات PUSH, POP

و هي مشروحة في الجدول التالي:

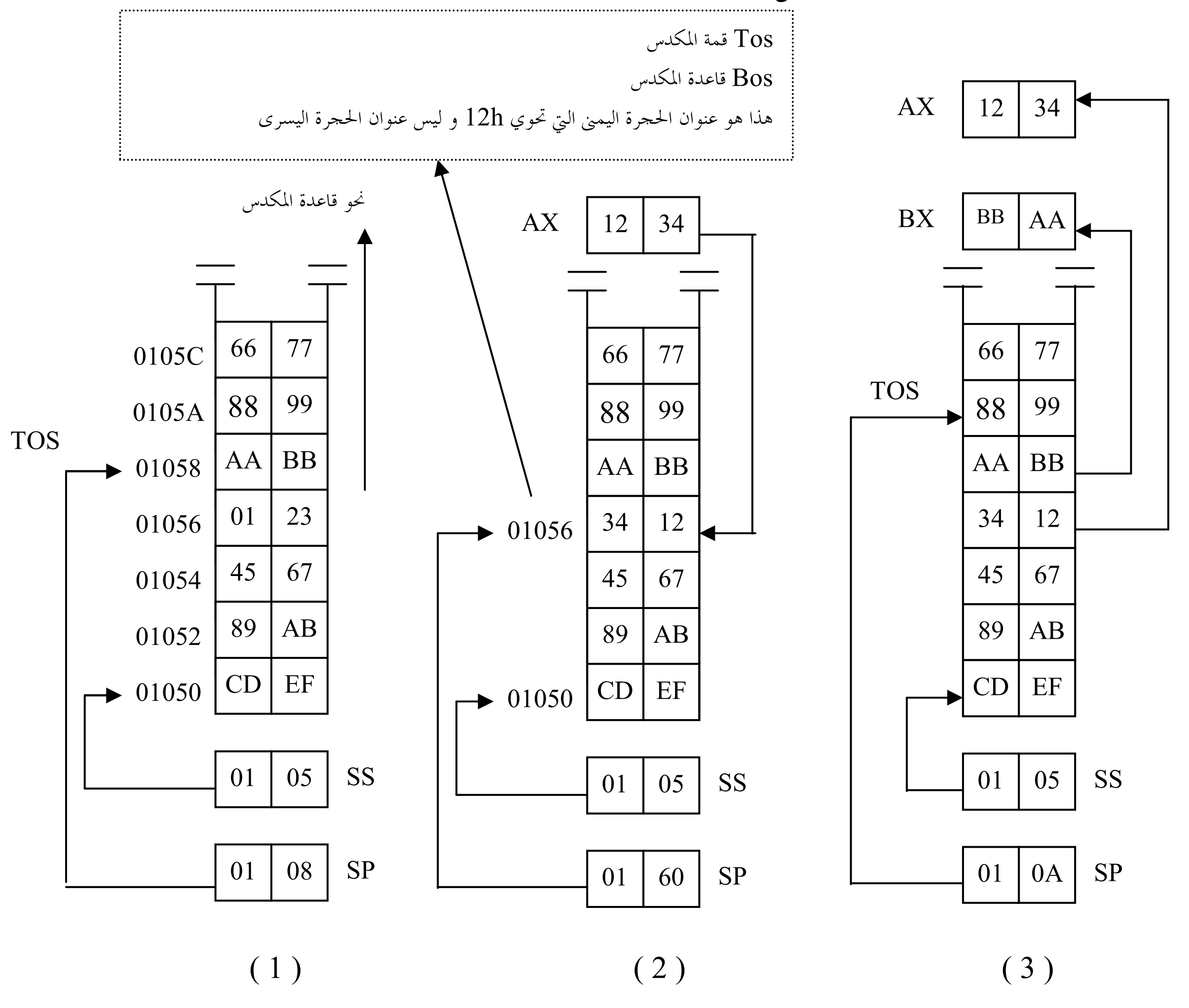
الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	$S \rightarrow ((SP))$	PUSH S	دفع كلمة إلى المكدس	PUSH
لا يوجد	$((SP)) \rightarrow D$	POP D	سحب كلمة من المكدس	POP

المكدس، مسجل مقطع المكدس SS ،مؤشر المكدس SP

أثناء عمليات المقاطعة ومناداة البرنامج الفرعي يتم دفع محتويات المسجلات الداخلية المعّينة بالمعالج إلى قسم من الذاكرة يدعى بالمكدس حيث تبقى هذه المحتويات هناك بشكل مؤقت. وعند إكمال روتين خدمة المقاطعة أو البرنامج الفرعي يتم سحب هذه القيم من المكدس وتوضع في نفس المسجل الداخلي حيث كان يحتوها أصلاً. فمثلاً عندما تحدث المقاطعة فإن المعالج و بشكل

أوتوماتيكي يدفع بمسجل الأعلام، القيمة الحالية في CS ، و القيمة الحالية في IP إلى المكدس. يمكن الحصول على مقطع مكدس جديد ببساطة بعنونة SS برمجياً من جديد. و إن مؤشر المكدس SP يحتوي على العنوان الفعال نسبة للقيمة في SS . و العنوان المشتق من محتويات SS و SP هو العنوان الفيزيائي لحجرة التخزين الأخيرة في المكدس (قمة المكدس) التي تمَّ دفــع المعطيـات إليها. إن القيمة في مؤشر المكدس تبدأ بـ FFFFh عند بدء تشغيل المعالج. و إن جمع هذه القيمة مع القيمة الحالية الموجودة في SS يعطي الحجرة ذات العنوان العلوي في المكدس (قاعدة المكدس). بما أن المعطيات المنقولة من و إلى المكدس عادة هي كلمات فإننا نتصور المكدس على شكل حجرات ذات 2 بايت، كما أنه من الضروري أن تكون جميع حجرات المكــدس في حــدود الكلمات الزوجية و ذلك لإنقاص عدد دورات الذاكرة المطلوبة لدفع أو سحب المعطيات من المكدس. يقــوم المعــالج بــدفع المعطيات و العناوين إلى المكدس كلمة في كل مرة، و في كل مرة يتم دفع قيمة مسجل ما إلى قمة المكدس فإن القيمة في مؤشر المكدس أولاً تنقص بمقدار 2 و من ثم تُكتب محتويات ذلك المسجل في ذاكرة المكدس. بهذه الطريقة فإن المكدس ينمــو نحــو الأسفل في الذاكرة انطلاقاً من قاعدة المكدس التي تطابق العنوان الفيزيائي المشتق من SS و القيمة FFFFh إلى نهايــة (قمــة) المكدس و التي تطابق العنوان الفيزيائي المشتق مت SS و العنوان الفعال 0000h و عندما تسحب القيمة من قمة المكدس فإن العكس لهذا التسلسل يحدث. إن العنوان الفيزيائي المعرف بواسطة SS و SP دائماً يشير إلى حجرة القيمة الأخيرة المدفوعة إلى المكدس حيث أن محتوياتها تسحب أولاً من المكدس إلى المسجل المعني ضمن المعالج ثم يزداد SP بمقدار 2 . إن قمــة المكــدس الجديدة تطابق القيمة السابقة المدفوعة إلى المكدس.

مثال: تبين الأشكال الثلاثة التالية حالات المكدس:



نلاحظ أن مسجل مقطع المكدس يحوي على 0105h و كما أشرنا سابقاً فإن قاعدة المكدس تكمن في العنوان الفيزيائي المشتق من SS مع العنوان الفعال FFFFh و هذا يعطي عنوان قاعدة المكدس BOS :

A (bos) = 0105h + FFFF = 1104Fh

بالإضافة إلى ذلك فإن مؤشر المكدس الذي يمثل العنوان الفعال من قاعدة المكدس إلى قمته يساوي 0008h لذلك فالقمة الحالية للمكدس هي في العنوان الفيزيائي:

A (tos) = 01050 + 0008 = 01058h

إن العناوين ذات القيم الأعلى من قمة المكدس 01058h تحتوي على معطيات حقيقية للمكدس بينما المعطيات ذات العناوين الأدنى من قمة المكدس ليست معطيات حقيقية للمكدس (بالتعريف : المكدس هو القيم المحصورة بين القاعدة و القمة) . نلاحظ أن القيمة الأخيرة المدفوعة إلى المكدس في الشكل الأول من الشكل السابق هي BBAAh . و يبين الشكل الثاني ما الذي يحدث عند تنفيذ تعليمة PUSH AX . هنا نجد أن محتويات AX هي 1234h و أن تنفيذ تعليمة PUSH يسبب إنقاص محتويات SP بمقدار 2 و لكنها لا تؤثر على محتويات مسجل مقطع المكدس SS لذلك فإن الحجرة التالية التي يتم الوصول إليها في المكدس تقابل العنوان 01056h . إلى هذه الحجرة يتم دفع القيمة المخزنة في AX إلى المكدس. نلاحظ أن البايت العلوي من المسجل AX (و الذي قيمته تساوي 12h) يكمن الآن في البايت السفلي للكلمة في المكدس و كذلك فالبايت السفلي من المسجل AX (و الذي قيمته تساوي 34h) يكمن الآن في البايت العلوي للكلمة في المكدس.

يبين الشكل الثالث ما الذي يحدث عندما تُسحب المعطيات من المكدس إلى المسجل الذي دُفعَت المعطيات منه إلى المكدس و ذلك بعد تنفيذ التعليمة POP AX ثم POP BX على الترتيب. نفس المناقشة بالنسبة إلى دفع قيمة فورية إلى المكدس.

عاشراً – تعليمات الحلقات

هناك ثلاث تعليمات مصممة بشكل خاص لتحقيق عملية الحلقة. و هذه التعليمات يمكن استعمالها بدلاً من تعليمات القفز الشرطي. و هي مبينة في الجدول التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		LOOP short_label	حلقة	LOOP

إنقاص CX بمقدار واحد دون التأثير على الأعلام ثم القفز إلى الحجرة المعرَّفة بواسطة اللافتة القصيرة إذا كان CX لا يساوي الصفر و إلا يتم تنفيذ التعليمة التالية لتعليمة الحلقة. و هنا يكون IP = IP + disp حيث disp أخذناها بعد تمديد إشارتها (أي جعلها بـ 16 بت).

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		LOOPE/ LC } لافتة قصيرة	حلقة طالما يساوي/ أو طالما صفر	LOOPE/ LOOPZ

إنقاص CX بمقدار واحد دون التأثير على الأعلام ثم القفز إلى الحجرة المعرَّفة بواسطة اللافتة القصيرة إذا كان CX لا يساوي الصفر و ZF يساوي الصفر و إلا يتم تنفيذ التعليمة التالية لتعليمة الحلقة.

و هنا جسم الحلقة فقط هو الذي يؤثر على الأعلام.

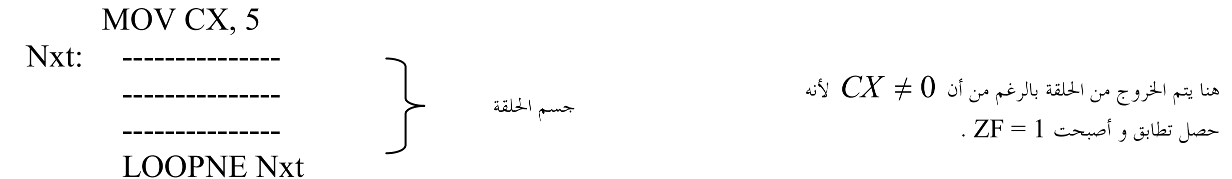
الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		LOOPNE/ LOOP } لافتة قصيرة	حلقة طالما لا يساوي/ أو طالما ليس صفراً	LOOPNE/ LOOPNZ

إنقاص CX بمقدار واحد ثم القفز إلى الحجرة المحددة بواسطة اللافتة القصيرة إذا كان CX لا يساوي الصفر و ZF يساوي الصفر و إلا يتم تنفيذ التعليمة التالية لتعليمة الحلقة.

و هنا أيضاً جسم الحلقة فقط هو الذي يؤثر على الأعلام.

مثال:

نريد البحث عن عنصر ضمن متجهة من العناصر (مصفوفة أحادية البعد) مثلاً : 8,9,4,5,7 و العنصر المراد إيجاده هو 4 . هنا CX = 5 و هو عدد العناصر. و يكون جسم الحلقة كالتالي:



11 – تعليمات السلسلة

نقصد بكلمة السلسلة أن بايتات أو كلمات معطيات تكمن في حجرات متعاقبة للذاكرة. إن تعليمات السلسلة تسمح للمبرمج بتنفيذ عمليات مثل نقل المعطيات من بلوك ذاكرة إلى بلوك آخر في الذاكرة، مسح أو كنس SCAN سلسلة من عناصر المعطيات المخزنة في الذاكرة و البحث عن قيمة معينة، مقارنة عناصر سلسلتين لتحديد فيما إذا كانا متطابقتين أو مختلفتين.

و تعليمات السلسلة الأساسية هي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		MOVS operand	نقل عنصر من سلسلة	MOVS

العملية: العنصر المحدد بواسطة DS:SI يتم نقله إلى الحجرة المحددة بواسطة القيمة ES:DI ثم:

$SI \pm 1 \text{ or } 2 \rightarrow SI$

$DI \pm 1 \text{ or } 2 \rightarrow DI$

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	نفس العملية السابقة و مقدار التزايد هو 1	MOVSB	نقل عنصر بايت من سلسلة	MOVSB
لا يوجد	نفس العملية السابقة و مقدار التزايد هو 2	MOVSW	نقل عنصر كلمة من السلسلة	MOVSW
أعلام الحالة		CMPS operand	مقارنة عنصر سلسلة	CMPS

العملية: يتم طرح متحول الهدف من متحول المصدر و لا تُخزن النتيجة إنما تُعدل أعلام الحالة فقط، أي:

((DS x 10h) + SI) – ((ES x 10h) + DI) → أعلام الحالة
SI ± 1 or 2 → SI
DI ± 1 or 2 → DI

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
أعلام الحالة		SCAS operand	مسح عنصر سلسلة	SCAS(B or W)

العملية :

(AL or AX) – ((ES x 10h) + DI) → أعلام الحالة
DI ± 1 or 2 → DI

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		LODS operand	تحميل عنصر سلسلة	LODS (B or W)

العملية :

((DS x 10h) + SI) → (AL or AX)
SI ± 1 or 2 → SI

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		STOS operand	تخزين عنصر سلسلة	STOS(B or W)

العملية :

(AL or AX) → ((ES x 10h) + DI)
DI ± 1 or 2 → DI

12ً – تعليمات تكرار السلسلة

في معظم التطبيقات يجب تكرار العمليات الأساسية للسلسلة من أجل معالجة جميع عناصرها. و يتم إنجاز هذا العمل بواسطة إدخال تعليمات التكرار قبل التعليمة الأساسية للسلسلة التي سوف تُكرر. هذا و إن أنواع تعليمات التكرار مبينة في الجدول التالي:

الكلمة المختزلة	المعنى	الاستخدام
REP	التكرار طالما لم نصل إلى نهاية السلسلة أي $CX \neq 0$	MOVS, STOS
REPE/REPZ	التكرار طالما لم نصل إلى نهاية السلسلة و السلسلتان متساويتان أي ZF=1, CX≠0	CMPS, SCAS
REPNE/REPNZ	التكرار طالما لم نصل إلى نهاية السلسلة و السلسلتان غير متساويتان أي ZF=0, CX≠0	CMPS, SCAS

مثال:

بفرض أن :

SI = 0100h　　DS = 0200h
DI = 0110h　　ES = 0400h

فإن نتيجة تنفيذ التعليمتين التاليتين :

```
MOV CX,20h
REP MOVSB
```

هي أن التعليمة الأولى تقوم بتحميل المسجل CX بالقيمة 32d = 20h أما التعليمة الثانية فتنقل 32 بايت من حجرات ذاكرة المصدر المحددة بواسطة DS و SI إلى بلوك حجرات ذاكرة الهدف المحددة بواسطة ES و DI .

13ً – تعليمتا مسح و توضيع علم الاتجاه

ذكرنا أنه يتم زيادة أو إنقاص قيم SI و DI بشكل أوتوماتيكي أثناء تنفيذ تعليمات السلسلة و أنه يتم تقرير الزيادة أو الإنقاص اعتماداً على قيمة علم الاتجاه DF حيث عندما DF = 0 تحدث الزيادة الأوتوماتيكية و العكس بالعكس. و يتم التحكم بعلم الاتجاه بواسطة التعليمتين التاليتين:

الكلمة المختزلة	المعنى	الصيغة	العملية	الأعلام المتأثرة
CLD	تنظيف DF	CLD	$0 \rightarrow DF$	DF
STD	توضيع DE	STD	$1 \rightarrow DF$	DF

14ً - تعليمتا IN و OUT

العملية	الصيغة	المعنى	الكلمة المختزلة
(port) → Acc	IN Acc,port	تعليمة دخل مباشرة	IN
((DX)) → Acc	IN Acc,DX	تعليمة دخل غير مباشرة	IN
Acc → (port)	OUT port,Acc	تعليمة خرج مباشرة	OUT
Acc → ((DX))	OUT DX,Acc	تعليمة خرج غير مباشرة	OUT

حيث في التعليمة المباشرة يكون طول الــ port بايتاً واحداً و في التعليمة غير المباشرة يكون DX محتوياً على عنوان نافذة.

مثال:

بفرض أن نافذتي دخل بحجم بايت في العناوين AAh, A9h على الترتيب ستُقرأ و من ثم سيتم إخراج محتوياتها إلى نافذة خرج بحجم كلمة في العنوان B000h المطلوب كتابة التعليمات اللازمة لإنجاز هذا العمل.

الحل:

```
IN AL,[0AAh]
MOV AH,AL
IN AL,[0A9h]
MOV DX,0B000h
OUT DX,AX
```

تمت بحمد الله